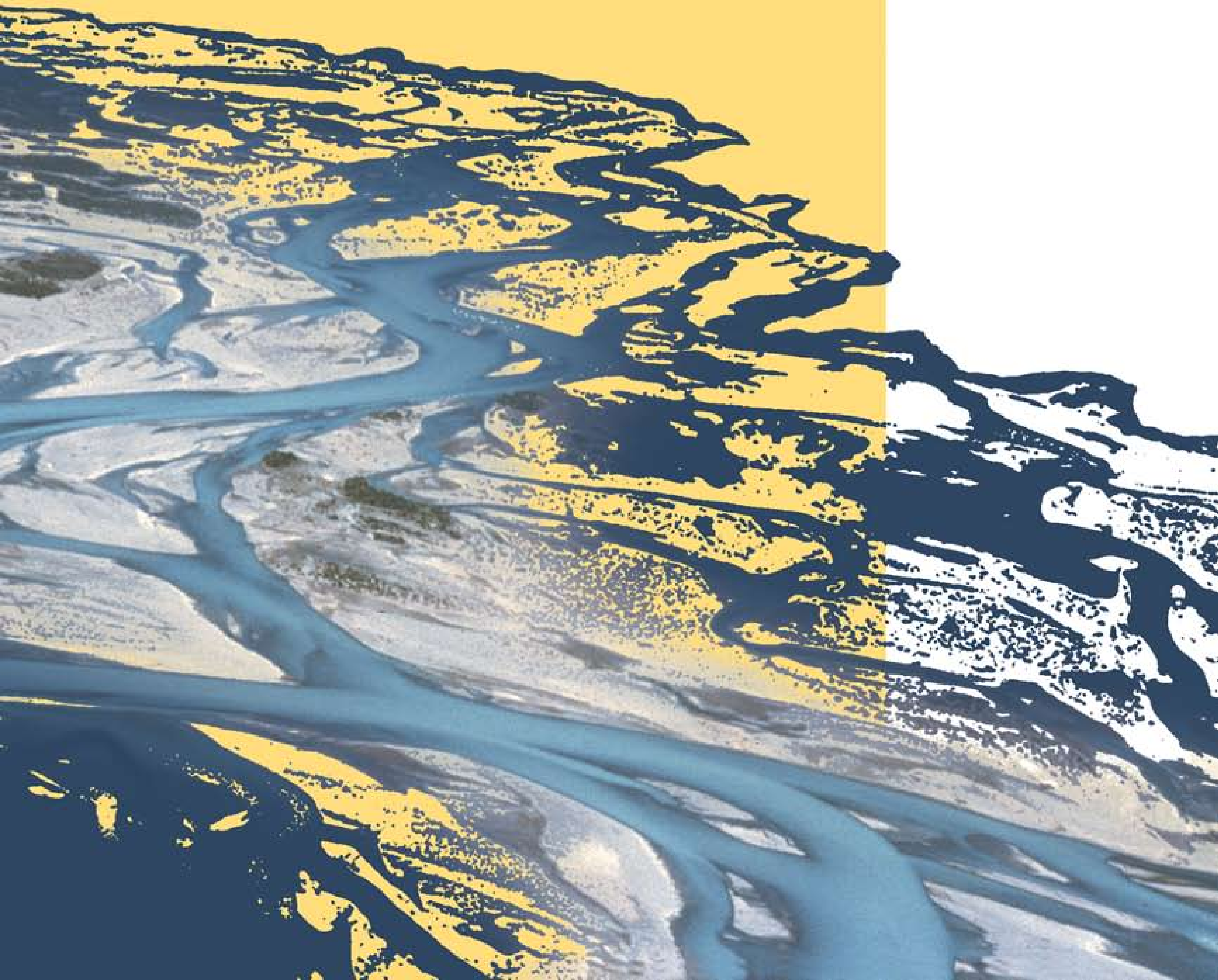
SYSTEM MANUALS

of BASEMENT



This page has been intentionally left blank.

VERSION 2.2
August, 2011

Project Team

Prof. Dr. R. Boes
Committee Member of Project "Integrales Flussgebietsmanagement", Director VAW

Dr. R. Fäh, Dipl. Ing. ETH
Committee Member of Project "Integrales Flussgebietsmanagement", Scientific Supervisor

R. Müller, Dipl. Ing. EPFL, Software Development, Scientific Associate
P. Rousselot, Dipl. Rech. Wiss. ETH, Software Development, Scientific Associate
C. Volz, Dipl. Ing., Software Development, Scientific Associate
L. Vonwiller, MSc. ETH, Documentation and Test, Scientific Associate
D. Vetsch, Dipl. Ing. ETH, Project Supervisor, Scientific Associate

Art Design and Layout

W. Thürig, D. Vetsch

Former Project Members

em. Prof. Dr.-Ing. H.-E. Minor,
Member of the steering committee of Rhone-Thur Project 2002-2007
Director of VAW 1998-2008

Dr. R. Veprek, Dipl. Rech. Wiss. ETH, Software Development, Sc. Associate, 2009-2010
Dr.-Ing. D. Farshi, MSc., Software Development, Scientific Associate, 2002-2007

Commissioned and co-financed by

Swiss Federal Office for the Environment (FOEN)

Contact

basement@ethz.ch
http://www.basement.ethz.ch

VAW
Laboratory of Hydraulics,
Hydrology and Glaciology

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Citation Advice

For System Manuals:

Faeh, R, Mueller, R, Rousselot, P, Vetsch, D, Volz, C., Vonwiller, L., Veprek R., Farshi, D 2011. *System Manuals of BASEMENT*, Version 2.1. Laboratory of Hydraulics, Glaciology and Hydrology (VAW). ETH Zurich. Available from <http://www.basement.ethz.ch>. [date of access].

For Website:

BASEMENT – Basic Simulation Environment for Computation of Environmental Flow and Natural Hazard Simulation, 2011. http://www.basement.ethz.ch

For Software:

BASEMENT – Basic Simulation Environment for Computation of Environmental Flow and Natural Hazard Simulation. Version 2.2. © VAW, ETH Zurich, Faeh, R., Mueller, R., Rousselot, P., Vetsch, D., Volz, C., Vonwiller, L., Veprek R., Farshi, D., 2006-2011.

Preface to Versions 1.0 – 1.3

The development of computer programs for solving demanding hydraulic or hydrological problems has an almost thirty-year tradition at VAW. Many projects have been carried out with the application of "home-made" numerical codes and were successfully finished. The according software development and its applications were primarily promoted by the individual initiative of scientific associates of VAW and financed by federal instances or the private sector. Most often, the programs were tailored for a specific application and adapted to fulfil costumer needs. Consequently, the software grew in functionality but with little documentation. Due to limited temporal and personal resources to absolve an according project, a single point of knowledge concerning the details of the software was inevitable in most of the cases.

In 2002, the applied numerics group of VAW was invited by the Swiss federal office for water and geology (BWG, nowadays Swiss Federal Office for the Environment FOEN) to offer for participation in the trans-disciplinary "Rhone-Thur" project. With the idea to build up a new software tool based on the knowledge gained by former numerical codes - while eliminating their shortcomings and expanding their functionality - a proposal was submitted. The bidding being successful a partnership in terms of co-financing was established. By the end of 2002, a newly formed team took up the work to build the so-called "BASic EnvironMENT for simulation of environmental flow and natural hazard simulation – BASEMENT".

From the beginning, the objectives for the new project were ambitious: developing a software system from scratch, containing all the experience of many years as well as state-of-the-art numerics with general applicability and providing the ability to simulate sediment transport. Additionally, professional documentation is a must. As to meet all these demands, a part wise reengineering of existing codes (Floris, 2dmb) has been carried out, while merging it with modern and new numerical approaches. From a software-technical point of view, an object-oriented approach has been chosen, with the aim to provide reusability, reliability, robustness, extensibility and maintainability of the software to be developed.

After four years of designing, implementing and testing, the software system BASEMENT has reached a state to go public. The documentation at hand confirms the invested diligence to create a transparent software system of high quality. The software, in terms of an executable computer program, and its documentation are available free of charge. It can be used by anyone who wants to run numerical simulations of rivers and sediment transport – either for training or for commercial purposes.

The further development of the software tends to new approaches for sediment transport simulation, carried out within the scope of scientific studies on one hand side. On the other hand, effectiveness and composite modelling are the goals. On either side, a reliable software system BASEMENT will have to meet expectations of the practical engineer and the scientist at the same time.

Minor

em. Prof. Dr.-Ing. H.-E. Minor
Member of the steering committee of Rhone-Thur Project 2002-2007
Director of VAW, 1998-2008

October, 2006

Preface to Versions 1.4

The work since the first release of the software in October 2006 was exciting and challenging. To go public is paired with interests and demands of users – although user support for the software never was intended. But interchange with users is definitely one of the most crucial factors of successful software development. Feedback from academic or professional users conveys a different point of view and enables the development team to achieve costumer proximity as well as to consolidate experience. Accordingly, the project team tried to meet the demands as effectively as possible. In version 1.3 of BASEMENT, which was released in April 2007, there were some errors fixed, a few new features added and the documentation was completed. Since then, many things have changed: on the personnel, on the project as well as on the software technical level.

In summer 2007 one of our main software developers, Dr. Davood Farshi, left VAW and changed to an international hydraulic consultant. Dr. Farshi supported our team from 2002 to 2007 as a profound numeric specialist and was mainly involved in the development of BASEplane. At his own request, he is still engaged in the development of BASEMENT as external advisor and tester. Dr. Farshi's position in the project team was reoccupied by Christian Volz, an environmental engineer from southern Germany. Mr. Volz has broad experience in numerical modelling as well as object-oriented programming.

On the project level the framework slightly changed. The initial scope within BASEMENT was developed, the "Rhone-Thur" project, has been finalized by the end of 2007. The sequel is called "Integrales Flussgebietsmanagement". It has the same co-financer as its predecessor, namely the Swiss federal office for the environment (FOEN), and basically the same participating institutions (EAWAG, WSL, LCH(EPFL) and VAW(ETHZ)). The funding runs until the end of 2011. Due to the retirement of Prof. Dr.-Ing. H.-E. Minor in summer 2008, our laboratory is solely represented in the project committee by Dr. R. Fäh at the moment.
The emphases of the new proposal for the further development of BASEMENT are advanced topics of hydraulics and sediment transport, such as secondary currents and lateral erosion. Furthermore, the efficiency of the software should be increased by the implementation of appropriate parallelisation and coupling approaches.

Since the last minor release a long time passed, which was mainly consumed by a general revision of the software. After five years of development a diligent consolidation was expedient. In addition, the coincidence of a new team member offered an unbiased reflection of the source code. All in all it was very worthwhile.
Last but not least, there are numerous bugs fixed and some new features in the current version. Mainly the efficiency of the software has been improved. The first stage of parallelisation is completed. The current implementation of the code includes the OpenMP interface which allows for parallel execution of the basic computation loops. In other words, the software is now able to exploit the power of current multi-core processors with a

convincing speedup. Furthermore, the revision of some data structures and output routines as well as the application of an optimised compiler led to a reduction in execution time.
Concerning sediment transport, the one-dimensional model BASEchain now supports the modelling of fine material, either as suspended or bed load. Also the advanced models for boundary conditions are worth mentioning. On the one hand, it is now possible to model domain boundaries with momentum and on the other hand, special boundary conditions inside the computational region, such as a weir or a gate, are implemented.
The fact, that the version 1.4 of BASEMENT is also available for the Linux operating system the first time, rounds off the new additions and features of the software package at hand.

Summarised one may say that the release 1.4 of BASEMENT is a major release due to all the different kinds of changes, but it's still a minor release concerning the new features – let's call it a "major minor" release. We are looking forward to Version 2.0 of BASEMENT, which is planned for next year.

D. Vetsch
Project Supervisor

October, 2008

Preface to Version 2.0

Four years ago, in spring 2006, the first version of the software system BASEMENT was completed and ready for internal use. In autumn of the same year, the first official version 1.1 of the software was released and made available as free download on the project website www.basement.ethz.ch. Since then, the functionality of the program has been enhanced and the international user community has grown gradually. Over the last years, BASEMENT has become a reliable tool for professional investigations, especially within the scope of flood prevention, and for scientific studies. Furthermore, the software is part and parcel of the lecture "Numerical Models in Hydraulic Engineering" to ensure education of young engineers in the field of hydrodynamic numerical simulation. The lecture is held on a regular basis by VAW staff for master students of civil and environmental engineering at ETH Zurich.

In February 2009, I have become the successor of Prof. em. Dr.-Ing. H.-E. Minor as Director of the Laboratory of Hydraulics, Hydrology and Glaciology (VAW) at ETH Zurich. In the meantime, I have joined the project committee "Integrales Flussgebietsmanagement"as a further representative of VAW besides Dr. R. Faeh.
Furthermore, there are some changes concerning the personnel of the project team of BASEMENT to mention. Lukas Vonwiller joined the team last autumn after having obtained his master's degree at ETH Zurich. Within the scope of his master thesis at the VAW, he studied the hydrodynamics and ecological impact of floods at the river Flaz using BASEMENT. Some of his experiences with the application of BASEMENT and selected results are documented in the new tutorial on 2-D simulations in the user manual UIV. His current duties are the application and testing of the software in terms of project work.
We were also very lucky being able to engage Dr. Ratko Veprek as a distinguished software engineer for a limited period of time. His contributions to the software, such as flow control of river systems, computational efficiency and the graphical user interface, just to name a few, are of great value. Unfortunately he will leave us by the day of the release to take on a post doctoral position abroad.

According to the announcement in the preface to version 1.4, the second major version of BASEMENT is released with little delay but with all the more important improvements and substantial new features. First of all, the new version 2.0 of the program comes with a graphical user interface (GUI), which allows running or stopping simulations and tracking the progress. Furthermore, the model setup and configuration, i.e. the assembling of the command file, is completely integrated into the GUI. The user is guided through the setup and any input is validated directly. In addition, the integrated help function, which is based on the command file reference, provides detailed information on the meaning of input parameters. This gives way to a clearer model setup compared to the rather fault-prone manual text editing, which is still available and also accessible through the GUI. Another main feature of the new GUI is the editing of the topography for BASEchain. Besides the GUI based setup, interpolation and thinning out of model cross sections, a graphical viewer helps the user to check the configuration and subdivision. For this reason, the new version

of BASEMENT comes with its own topography file format for BASEchain. The new format has a clear structure similar to the style of the command file.
Moreover, the visualisation of actual results during a simulation with BASEviz has been improved and is now more interactive, i.e. the simulation can be paused, continued or the variable shown can be switched. Other improvements concern computational efficiency and sediment transport, especially gravitational bed load transport. Please refer to the release notes in the section "introduction and installation" of this manual for further details about new features and bug fixes.

The software system BASEMENT in its current version 2.0 has reached the point to be termed as a state of the art numerical modelling tool for flow and sediment transport in rivers. The incorporated well established or new numerical approaches, software technical features like parallelization or the coupling of sub domains, advanced features for sediment transport and flow control are making it a reliable tool for professional as well as scientific applications. With the new GUI another hurdle has been cleared and a new era of the software in terms of usability has begun. We are looking forward to the further development as well as upcoming releases of BASEMENT and we are curious about how the software will establish itself in the future.

[illegible]

Prof. Dr. R. Boes
Committee Member of Project "Integrales Flussgebietsmanagement"
Director of VAW

May, 2010

SOFTWARE LICENSE

between

ETH Zurich
Rämistrasse 101
8092 Zürich
Represented by Prof. Dr. Robert Boes
VAW
(Licensor)

and

Licensee

1. Definition of Software

The Software system BASEMENT is composed of the executable (binary) file BASEMENT and its documentation files (System Manuals), together herein after referred to as “Software”. Not included is the source code.
Its purpose is the simulation of water flow, sediment and pollutant transport and according interaction in consideration of movable boundaries and morphological changes.

2. License of ETH Zurich

ETH Zurich hereby grants a single, non-exclusive, world-wide, royalty-free license to use Software to the licensee subject to all the terms and conditions of this Agreement.

3. The scope of the license

a. Use

The licensee may use the Software:
- according to the intended purpose of the Software as defined in provision 1
- by the licensee and his employees
- for commercial and non-commercial purposes

The generation of essential temporary backups is allowed.

b. Reproduction / Modification

Neither reproduction (other than plain backup copies) nor modification is permitted with the following exceptions:

Decoding according to article 21 URG [Bundesgesetz über das Urheberrecht, SR 231.1)
If the licensee intends to access the program with other interoperative programs according to article 21 URG, he is to contact licensor explaining his requirement.
If the licensor neither provides according support for the interoperative programs nor makes the necessary source code available within 30 days, licensee is entitled, after reminding the licensor once, to obtain the information for the above mentioned intentions by source code generation through decompilation.

c. Adaptation

On his own risk, the licensee has the right to parameterize the Software or to access the Software with interoperable programs within the aforementioned scope of the licence.

d. Distribution of Software to sub licensees

Licensee may transfer this Software in its original form to sub licensees. Sub licensees have to agree to all terms and conditions of this Agreement. It is prohibited to impose any further restrictions on the sub licensees’ exercise of the rights granted herein.

No fees may be charged for use, reproduction, modification or distribution of this Software, neither in unmodified nor incorporated forms, with the exception of a fee for the physical act of transferring a copy or for an additional warranty protection.

4. Obligations of licensee

a. Copyright Notice

Software as well as interactively generated output must conspicuously and appropriately quote the following copyright notices:

Copyright by ETH Zurich, VAW, Faeh R., Mueller R., Rousselot P., Vetsch D., Volz C., Vonwiller L., Veprek R., Farshi D., 2006-2011

THIRD PARTY SOFTWARE COPYRIGHT NOTICES

The Visualization Toolkit (VTK)

CVM Class Library

Qt v4.6 – Cross-platform application and UI framework

Qwt - Qt Widgets for Technical Applications

BASEMENT v2.2 is based in part on the work of the Qwt project (http://qwt.sf.net).

CGNS – CFD General Notation System

cgnslib 4.5 - http://cgns.sourceforge.net

U

USER MANUAL

of BASEMENT

This page has been intentionally left blank.

the

BASIC SIMULATION ENVIRONMENT

alias **BASEMENT**

This page has been intentionally left blank.

Table of Contents

1 General Use

1.1 Problem Description ... 1.1-1
1.2 Product Delineation and Employment Domains ... 1.2-1
1.2.1 Product Delineation ... 1.2-1
1.2.2 Employment Domains ... 1.2-1
1.3 Processed Data Types and Capabilities ... 1.3-1
1.3.1 Processed Data Types ... 1.3-1
1.3.2 Capabilities ... 1.3-1

2 System Overview

2.1 Numerical Subsystems ... 2.1-1

3 Components

3.1 Mathematical – Physical – Modules ... 3.1-1
3.2 Computational Grid ... 3.2-1
3.2.1 The Meta Model ... 3.2-1
3.2.2 BASEchain : one dimensional model ... 3.2-3
3.2.3 BASEplane : two dimensional model ... 3.2-4

4 Simulation Procedure

4.1 Flow of main activities ... 4.1-1
4.2 Scenario Examples ... 4.2-1
4.2.1 Sediment balance in a river 1-D ... 4.2-2
4.2.2 Flood 1-D + 2-D ... 4.2-3
4.2.3 Debris flow 2-D ... 4.2-3
4.3 Executing BASEMENT ... 4.3-1
4.3.1 Executing BASEMENT with graphical user interface (GUI) ... 4.3-1
4.3.2 Executing BASEMENT on Microsoft Windows ... 4.3-1
4.3.3 Executing BASEMENT on Linux ... 4.3-1
4.3.4 Executing BASEMENT in batch mode ... 4.3-1
4.3.5 Restart the simulation in BASEplane from a given solution ... 4.3-2

5 Parallelization

5.1 Overview ... 5.1-1
5.1.1 Needs for parallel computing ... 5.1-1
5.1.2 Parallel computer architectures ... 5.1-1
5.2 Parallelization issues on shared memory systems ... 5.2-1
5.2.1 Levels of parallelization ... 5.2-1
5.2.2 Factors influencing parallel performance ... 5.2-2

5.3 Parallelization with OpenMP 5.3-1
5.3.1 Overview 5.3-1
5.3.2 Characteristics of OpenMP 5.3-1
5.3.3 Parallel directives in OpenMP 5.3-2

6 Model Coupling

6.1 Introduction 6.1-1
6.2 Coupling Types 6.2-1
6.3 Coupling Mechanisms 6.3-1
Explicit coupling of sub-domains 6.3-1
6.3.1 One-way coupling and two-way coupling 6.3-1
6.4 Definitions of Exchange Conditions 6.4-1
6.4.1 General remarks 6.4-1
6.4.2 Exchange conditions for mixed-dimensional sub-domains 6.4-2
6.4.3 Exchange conditions for river junctions in 1-D river networks 6.4-2
6.4.4 Exchange conditions for river bifurcations in 1-D river networks 6.4-3
6.4.5 Exchange conditions for combined 1-D and 2-D modelling 6.4-4
6.4.6 Data exchange for morphological simulations with multiple grain classes .. 6.4-5
6.5 Synchronization Concept 6.5-1
6.5.1 General remarks on synchronization 6.5-1
6.5.2 “Local time stepping” approach 6.5-1
6.6 External Coupling 6.6-1
6.6.1 Introduction 6.6-1
6.6.2 Data exchange over TCP/IP 6.6-2

7 Flow Control in River Systems

7.1 Introduction 7.1-1
7.2 Concept of Flow Control 7.2-1
7.2.1 Monitored Variable 7.2-1
7.2.2 Manipulated Variable 7.2-1
7.3 Controller Types 7.3-1
7.3.1 PID-Controller 7.3-1

List of Figures

Fig. 1: *System Overview* *2-1*

Fig. 2: *Modules and their Components* *3-1*

Fig. 3: *The Meta Model: fusion of "real world" data with abstract numerical considerations* .. *3.2-1*

Fig. 4: *Discrete Representation of the Topography within BASEchain* *3.2-3*

Fig. 5: *Discrete Representation of the Topography within BASEplane* *3.2-4*

Fig. 6: *Activity diagram generate computational mesh* *4.2-1*

Fig. 7: *Activity diagram Sediment balance in a river* *4.2-2*

Fig. 8: *Activity diagram Flood 1-D + 2-D* *4.2-3*

Fig. 9: *Activity diagram Debris flow 2-D* *4.2-3*

Fig. 10: *Shared memory architectures (left) and distributed memory architectures (right)* *5.1-2*

Fig. 11: *Iillustration of different levels of parallelism* *5.2-1*

Fig. 12: *Compiler directives for basic parallelization constructs in OpenMP* *5.3-2*

Fig. 13: *Critical sections and barriers for synchronization operations in OpenMP* *5.3-3*

Fig. 14: *River network with multiple BASEchain (1-D) and BASEplane (2-D) sub-domains and several coupling interfaces.* *6.1-1*

Fig. 15: *Modeling of a river junction with two different approaches (black arrows indicate a confluence of river branches, red arrows a bifurcation).* *6.4-3*

Fig. 16: *Conceptual overview of combined 1-D river flow and 2-D floodplain modeling* *6.4-4*

Fig. 17: *Mapping of grain compositions from one sub-domain to another* *6.4-6*

Fig. 18: *LTS-synchronization for 3 sub-domains with different time step sizes. The sub-domain C with the smallest time step size determines the base time step. Sub-domains A and B run for multiples of 4 and 2 of the base time step size.* *6.5-2*

Fig. 19: *Connection request from external program (client) to BASEMENT (server)* *6.6-2*

Fig. 20: *Sending and receiving data from socket stream (FIFO pipe)* *6.6-3*

Fig. 21: *Basic Control Cycle* *7.2-1*

List of Tables

Tab. 1: OpenMP parallelization –pro and contra 5.3-2

Tab. 2: Listing of different coupling types and their descriptions. 6.2-1

Tab. 3: Possible exchange conditions between sub-domains. 6.4-1

Tab. 4: Exchange conditions for mixed-dimensional coupling (i = index of 2-D edge or 2-D element) 6.4-2

Tab. 5: Exchange conditions for river junctions 6.4-3

Tab. 6: Exchange conditions for river bifurcations 6.4-4

Tab. 7: Exchange conditions for lateral coupling 6.4-5

I BASIC SIMULATION ENVIRONMENT

The software system „BASEMENT“ (basic-simulation-environment) shall provide a flexible and functional environment for numerical simulation of alpine rivers and sediment transport involved. The numerical models for the computation of one- and two dimensional flows with moving boundaries and appropriate models for bed load as well as suspended load are forming the core of the software system. Two of the main project tasks were the renewal and further development of the existing 1-D and 2-D models (“Floris”, “2dMB”). The one-dimensional model complies with the upper bound of the considered spatial scale (maximum idealization and slightest resolution of spatial processes) and is meant to provide appropriate boundary conditions for the 2-D and 3-D models.

The main focus of conception and development was the stability of the numerical models, the flexibility of the computational grid and the combination and efficiency of the method of calculation (problem dependent equations, coupling of models, parallelization).

The development process was orientated at the concepts of object orientation, to assure transparency, documentation and flexibility of the software system as far as possible. Future developments, applications related to practice and scientific projects shall build upon the environment of BASEMENT to ensure sustainability.

This page has been intentionally left blank.

1 General Use

1.1 Problem Description

In connection with watercourses and river areas, increasingly complex problems have to be addressed. The estimation of floods, the more frequent occurrence of restoration projects or the study of naturally shaped watercourses implicate the examination of larger regions - also outside of the actual waterway - and a more manifold shape of the channels. The simple formulas for the calculation of flow behaviour used in the past showed in several cases to be insufficient to obtain the desired information. The extent of the considered areas makes the application of hydraulic models in a laboratory - usually employed for difficult cases - impossible or too expensive. So, the numerical simulation of flow behaviour is in many cases the most obvious solution. However, existing programs have still some weak points. Some are limited in their capabilities (e.g. only steady flow, no sediment transport and one dimension only) or may lack in user support caused in incompleteness of documentation or training of users. Furthermore, inherent numerical problems request certain expertise to be overcome. In addition, the preparation of the input data and the processing of the results to a shape, which facilitates the interpretation, are often very laborious.

The aim of the software system BASEMENT, in terms of its free availability and its accompanying scholar programs, is to enable a broader range of people to skilfully process river modelling projects in a justifiable amount of time.

This page has been intentionally left blank.

1.2 Product Delineation and Employment Domains

1.2.1 Product Delineation

BASEMENT is a river engineering tool, which supports the engineer in the solution of tasks in the domain of river area modelling. The program permits reliable computations based on state of the art numerical tools, constant onward development and successive realisation of case studies.
Unlike currently used programs for the simulation of a specific flow behaviour, BASEMENT intends the arrangement of many different problem types with one single tool to gain a integrated understanding for the initial position, the solution process and its results.

1.2.2 Employment Domains

The aim of BASEMENT is to permit the solution of as many problems as possible in the domain of river engineering, especially in cases for which the traditional dimensioning tools are insufficient and studies including physical hydraulic models are not possible or too expensive.
Typical employment domains are:

- Several problems in relation with the sediment transport of water courses, for instance the future development of deltas and alluvial fans, the long term evolution of the bottom of channels, or the aggradations of storage spaces and the consequences of their scavenging;

- River engineering enterprises, which imply the modification of the channel geometry, as this can be the case for example for revitalisations or protection measures, where the consequences of the interventions have to be evaluated;

- Identification and quantification of dangers for the development of danger maps or of protection and emergency measures, considering the flow behaviour and sediment deposition both inside and outside of the main channel, as well as erosion danger, and consequences of debris flows and dam breaks.

This page has been intentionally left blank.

1.3 Processed Data Types and Capabilities

1.3.1 Processed Data Types

The raw data can be divided into three groups:

- **topographic data**, particularly elevation models and cross sections
- **hydrologic data**: time series of flow discharge, water levels or concentration of suspended sediments, velocity profiles;
- **granulometric data**: grain size distributions from water-, sediment- or line samples.

1.3.2 Capabilities

BASEMENT has the following fundamental capabilities:

- Simulation of flow behaviour under steady and unsteady conditions in a channel as well as its transition;
- Simulation of sediment transport (both bed load and suspended load) under steady and unsteady conditions in a channel with arbitrary geometry;
- Simulation of erosion;
- Choose between different approaches (e.g. choice of problem matched solver-algorithms);

This page has been intentionally left blank.

2 System Overview

At the current stage of development, the software system consists of the numerical subsystems and the different interfaces to the infrastructural software, such as pre- and post processors. The core of BASEMENT consists of the numerical solution algorithms comprised in the appropriate modules. Pre- and post-processing can be performed with independent products using a well defined common interface. The flexible software design enables a future adoption to a common database for input- and output data.

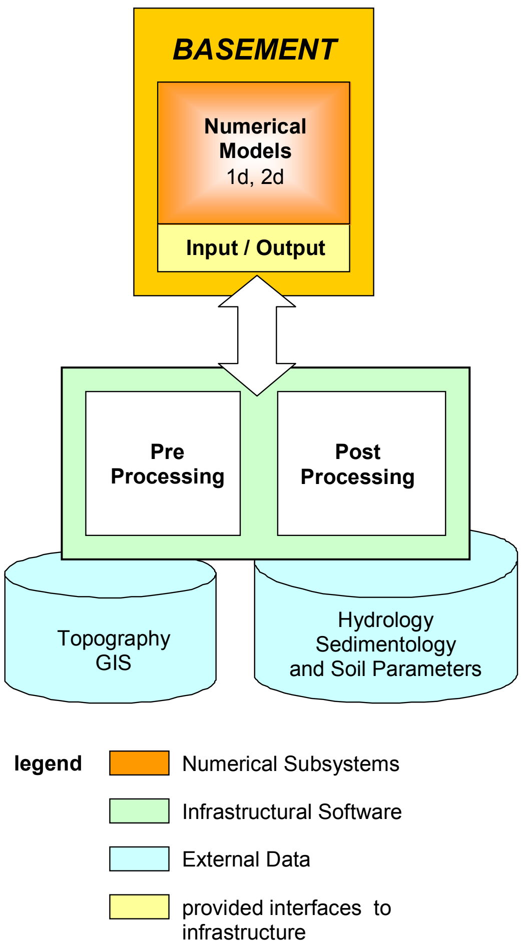


Fig. 1: System Overview

This page has been intentionally left blank.

2.1 Numerical Subsystems

The core of the software system consists of the numerical subsystems, which actually are:

BASEchain

The one dimensional numerical tool named *BASEchain* enables the simulation of river reaches (based on cross sections) with respect to sediment transport. A coupling with the 2-D tool is foreseen.

BASEplane

The two dimensional numerical tool named *BASEplane* enables the simulation of river reaches as well as flood plains (bases on a digital terrain model) with respect to sediment transport.

BASEspace (planned)

The three dimensional numerical tool named *BASEspace* is meant for the simulation of local flow fields (based on spatial geometry) with respect to sediment transport. A coupling with the 2-D tool is foreseen.

For a list of implemented features of each model, please refer to the actual release notes.

This page has been intentionally left blank.

3 Components

To reveal the “black box” of the numerical models, *Fig. 2* gives a graphical insight. The simulation tools of BASEMENT can be subdivided into three different parts:

- the mathematical-physical modules consisting of the governing flow equations
- the computational grid representing the discrete form of the topography
- and the numerical modules with their methods for solving the equations

In the next few chapters an overview of these modules is given. A more detailed description can be found in the reference manual.

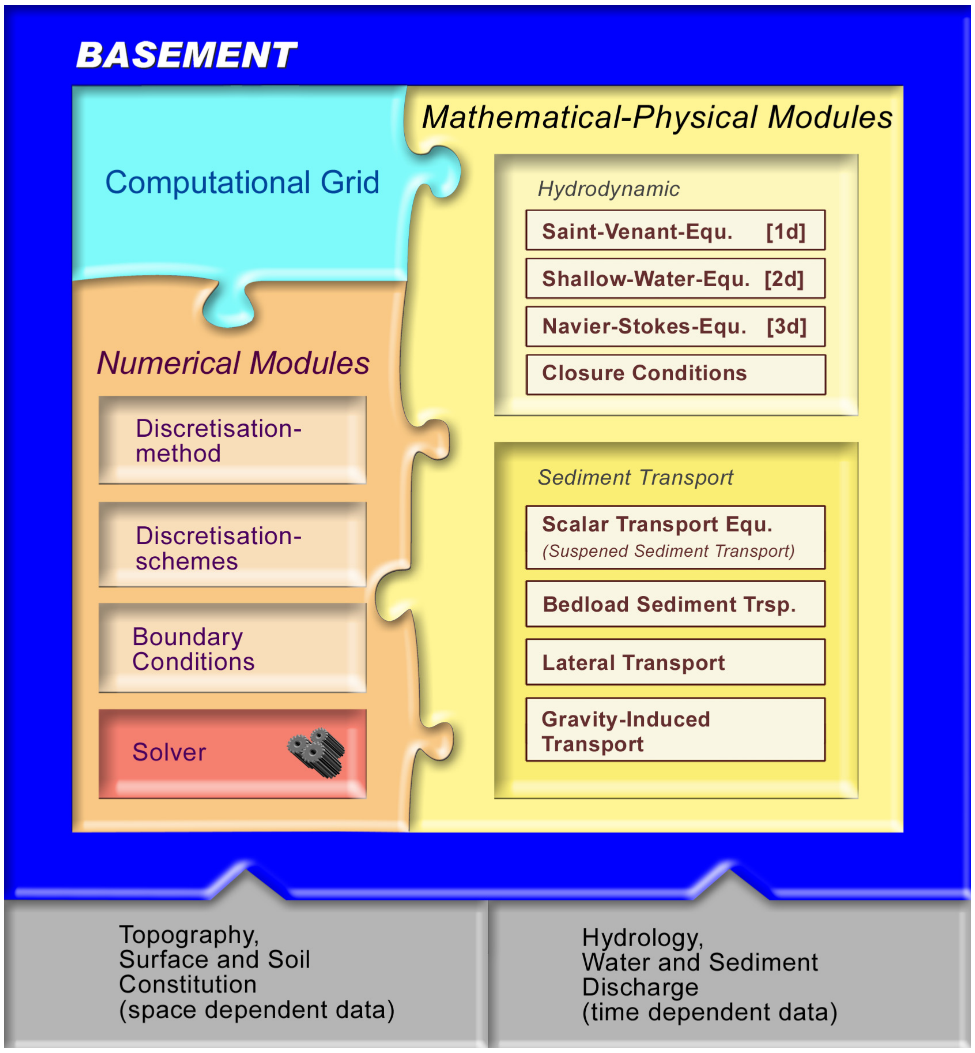


Fig. 2: Modules and their Components

This page has been intentionally left blank.

3.1 Mathematical – Physical – Modules

The behavior of the fluid hydraulics can be explained with physical models, namely the conservation of mass and momentum. Theoretically, it is possible to resolve the mathematical problem up to small scale phenomena like turbulence structures. In a natural problem however, it is mostly impossible to determine all boundary- and the exact initial conditions. Furthermore, the computational time needed to solve the full equation system is increasing very fast with higher spatial and temporal resolution. Therefore, dependent on the problem, simplified mathematical models are used.

In three dimensions, the flow and pressure distribution are completely described by the Navier-Stokes equations. These equations can only be solved numerically, as analytical solutions exist only for some strongly simplified problems. The 3-dimensional approach is only suitable for local problems, where turbulence phenomena and flow in all directions are essential for the results, e.g. the flow around bridge piers.

Assuming a static pressure distribution and neglecting the vertical flow components, the Navier-Stokes equations simplify to the 2-dimensional shallow water equations. This set of equations provides accurate results for the behaviour of water level and velocities in a plane. Turbulence effects cannot be resolved anymore but are accounted for by an artificial friction factor in the closure condition, which establishes a relation between flow velocity and shear stress. The shallow water equations are used for 2-dimensional flows like dam breaks, curved flow etc.

Reducing the spatial dimension once more results in the 1-D Saint-Venant equations. The main outputs of these equations are the water level and mean velocity in flow direction. This method is still in use for computing large river systems.

The computation of sediment transport is mathematically not as well developed as the hydrodynamic part. Theoretically, the movement of every single stone within the sediment could be computed by solving its equation of motion. However, this approach is yet numerically too expensive.
Therefore, sediment transport and behaviour of the riverbed are computed using empirical formulas developed by river engineers. The computation of the sediment flux is physically not really correct, but proved to be accurate enough for a broad range of sediment transport problems.
Usually, sediment transport occurs in the main flow direction. More sophisticated models consider also lateral phenomena within a curved flow.

Very small grain sizes are treated as suspended sediment load. Their behaviour can be computed by a physically scalar transport equation.

This page has been intentionally left blank.

3.2 Computational Grid

3.2.1 The Meta Model

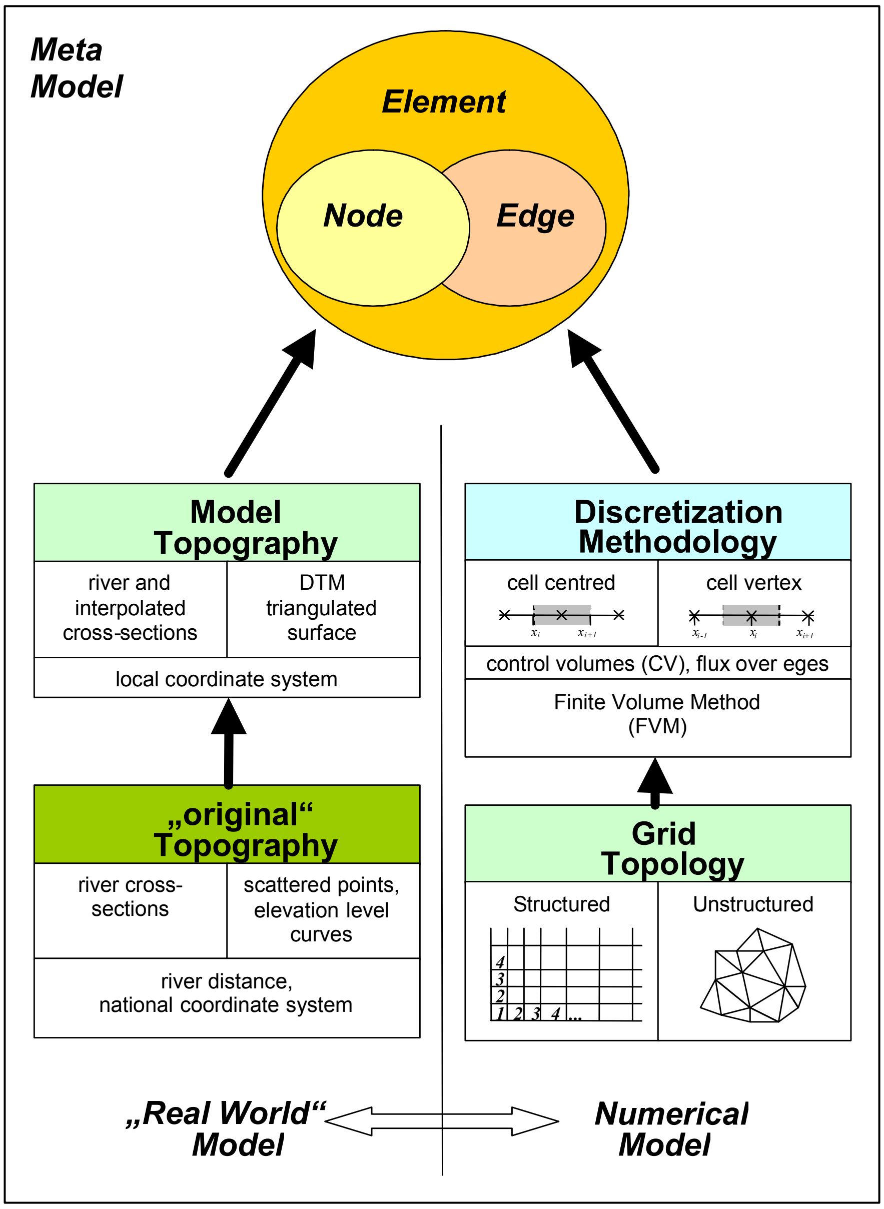


Fig. 3: The Meta Model: fusion of "real world" data with abstract numerical considerations

An important aspect of every computational task is the grid generation where the real world topography data is transformed into an internal computational grid on which the governing equations are solved. Independent of the discretization method, the construction of the computational grid has great impact on the accuracy of the results and on the computational time needed for the simulation or the numerical time step, respectively. Generally, a suitable mesh is dense at regions, where strong changes in the flow occur and coarse in regions of lower interest. Additionally, the grid cells should not underlie strong deformations.

Usually, the raw real world data comes in form of river cross sections or geographic terrain models e.g. from a GIS. This elevation information has to be mapped onto a suitable mesh.

There are two types of computational grids: structured and unstructured ones.
Structured grids consist of quadrilaterals and can be mapped onto a Cartesian domain. They allow for simple data structures and efficient algorithms. The mesh generation is relatively simple and can even be done manually. However, structured meshes are somehow unhandy for the representation of arbitrary topography data.
Unstructured grids are mostly composed of triangles and cannot be mapped onto Cartesian meshes. They usually need more complicated data structures but are highly flexible for automatic mesh generation in complex geometries. An unstructured grid is the most general case of a grid based discretization and is perfectly suitable for object oriented modelling. BASEMENT is built on unstructured grids.

The computational grid consists in general of cells (control volumes). In the software model, the mesh is based on three different objects:

- [node] the nodes – mass free points in relation to a coordinate system;
- [edge] the edges, which are defined by two nodes and define the place of information flux between two elements in Finite Volume Methods;
- [element] the elements, which are defined by several nodes and define the place of the physical variables, e.g. cell centered methods.

This data structure allows for similar treatment of 1-D and 2-D methods and schemes.

As the difficulty of mesh generation occurs in many different computational tasks, a broad range of different triangulation techniques or mesh refinement methods can be found in the literature. Some of them are specially designed for a certain discretization scheme but can also be used elsewhere.
As there is nothing such as an ideal or perfect mesh, the user is recommended to produce different grids and compare their behaviour to find the best solution. There are commercial tools available which can be used for the grid generation, e.g. SMS. However, the BASEMENT standard for grid representation (see Reference manual) also allows for self created meshes.

3.2.2 BASEchain : one dimensional model

Discrete Representation 1d	Overview

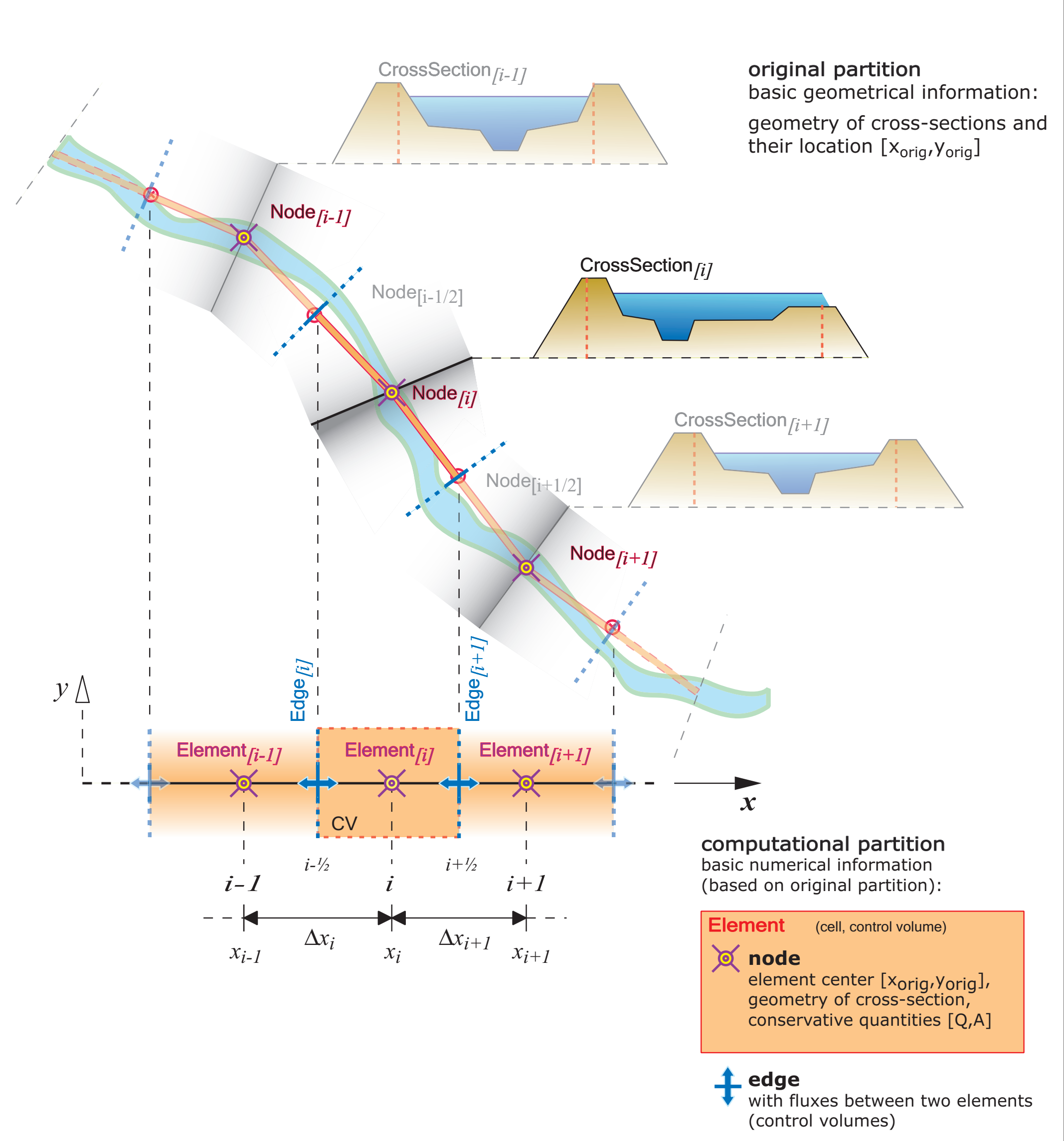

CrossSection[i-1]
original partition
basic geometrical information:
geometry of cross-sections and their location [x_orig,y_orig]
Node[i-1]
Node[i-1/2]
CrossSection[i]
Node[i]
CrossSection[i+1]
Node[i+1/2]
Node[i+1]
y
Edge [i]
Edge [i+1]
Element[i-1]
Element[i]
Element[i+1]
x
CV
i-½
i+½
i-1
i
i+1
Δx_i
Δx_i+1
x_i-1
x_i
x_i+1
computational partition
basic numerical information (based on original partition):
Element (cell, control volume)
node
element center [x_orig,y_orig], geometry of cross-section, conservative quantities [Q,A]
edge
with fluxes between two elements (control volumes)

Fig. 4: Discrete Representation of the Topography within BASEchain

In one dimension, an element consists of two nodes with known cross-section. With a cell-centred discretization, all variables – velocity, flow depth and cross-section geometry - are defined at the location of the nodes. The midpoint of the connecting line between two nodes defines the common edge of the two elements.
The more nodes are known, the better the representation of the real world data, especially at regions with strongly curved watercourse.

3.2.3 BASEplane : two dimensional model

Discrete Representation 2d	Overview

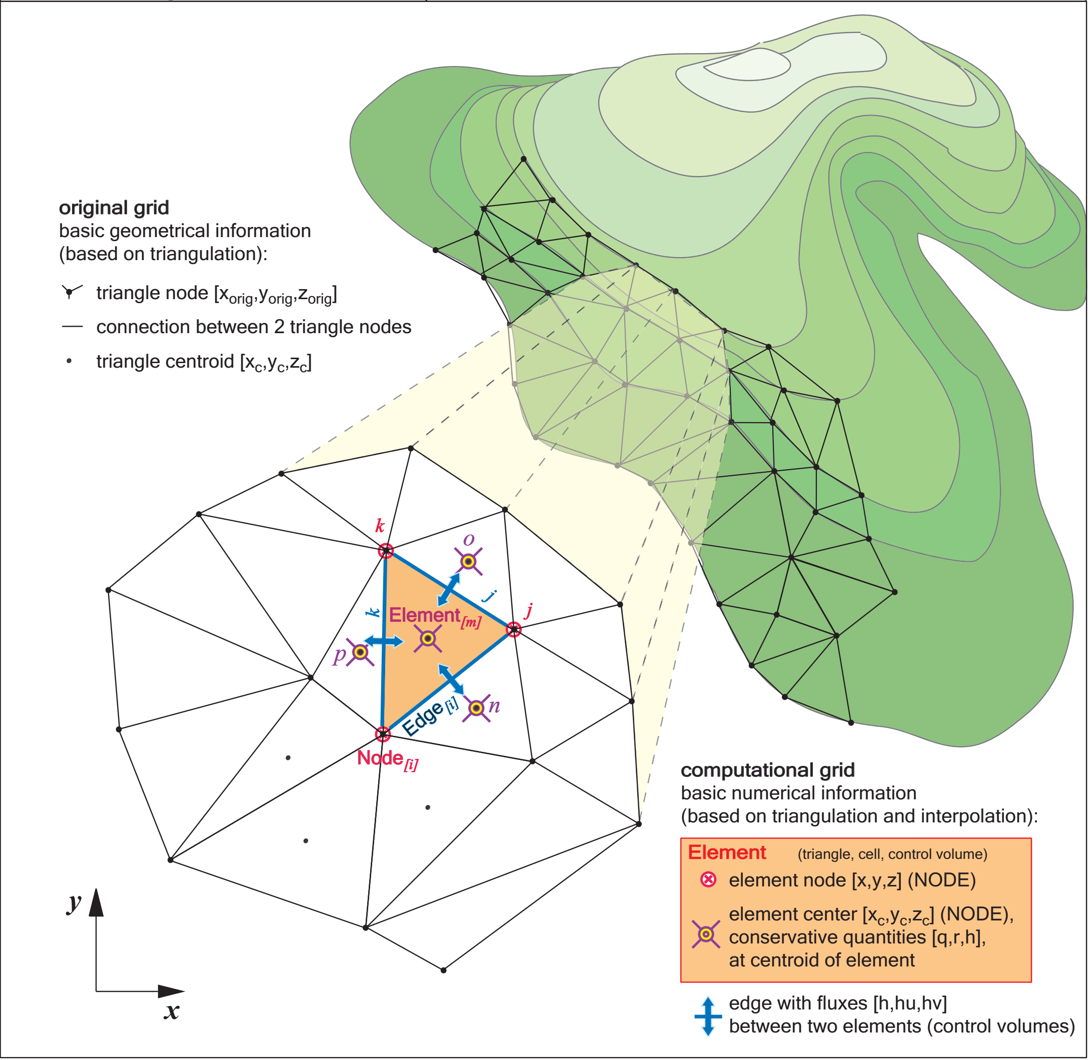

original grid
basic geometrical information
(based on triangulation):
triangle node [x_{orig},y_{orig},z_{orig}]
connection between 2 triangle nodes
triangle centroid [x_c,y_c,z_c]
k
o
j
k
Element [m]
j
p
n
Edge [i]
Node [i]
y
x
computational grid
basic numerical information
(based on triangulation and interpolation):
Element (triangle, cell, control volume)
element node [x,y,z] (NODE)
element center [x_c,y_c,z_c] (NODE),
conservative quantities [q,r,h],
at centroid of element
edge with fluxes [h,hu,hv]
between two elements (control volumes)

Fig. 5: Discrete Representation of the Topography within BASEplane

In two dimensions, an element consists of three nodes with a known ground elevation. Usually, this real world height information is not given exactly at the desired node coordinates and therefore has to be interpolated.
The primary variables are defined somewhere inside the element, e.g. the balance point. The fluxes between two elements are defined at their corresponding edges.

4 Simulation Procedure

The procedure to simulate a concrete problem setup is not unique. BASEMENT is coded using an object-oriented design which allows for flexibility and interchange ability concerning different application problems. The possible combinations are manifold.
On the one hand, the governing equations may change dependent on simplifications or extensions of certain terms, use of sediment transport or pure hydraulics, etc. On the other hand, there are miscellaneous numerical methods, e.g. for time integration (implicit, semi-implicit, explicit) or computation of spatial fluxes. Therefore, the main variables of interest differ from one problem to the other.
It is of great importance, to plan carefully each simulation approach to a certain problem. The most difficult and time-consuming part is not the simulation itself but the acquisition of all needed data (topography, boundary- and initial conditions) and a proper setup of this data.
This section describes the main activities performed to execute a simulation with BASEMENT in a very general case. In most problems, only a part of them are being used.

This page has been intentionally left blank.

4.1 Flow of main activities

1 **Build and Name Project**

2 **Preparative Work / Scenario definition**

- Topography / computational mesh
    - Import rough topographic data
    - Import cross sections
    - Add break lines manually
    - Build terrain model (triangulate)
    - Build cross sections from terrain model
    - Determine mean riverbed level
    - Generate computational mesh
- Define properties of mesh elements
    - Roughness
    - Grain sizes / distribution / mean diameters
    - Flow through / no flow through elements
    - Mobile / fixed bed elements
    - Transport relevant / not relevant elements
    - Shear stress
- Define approaches / Simulation Matrix
    - Sediment transport formulas
    - Roughness / Friction formulas
    - Debris flow type and approach

3 **Define initial- / boundary conditions and required results**

- Boundary conditions
    - Time discharge from Inlet Hydrographs
    - Water level time series at outflow
    - Special hydraulic elements (weir, step, bridge, block, wall, …)
    - Sediment discharge dependent on Hydrographs
    - Concentration of suspended material
    - Grain size distribution
    - Height of Subsurface
    - Viscosity
- Initial conditions
    - Waterlevel
    - Sediment bed level / Slope
    - Flow velocities
    - Grain size distribution
- Required results
    - Monitoring point or sections
    - Values: discharge, velocity, water/bed-level, concentration, shear stress, …
    - Result types: max (t), min (t), integral, time series with designed timestep, Simulation times, error estimation, …

4 **Execute Simulation**

5	**Elaborate results** ▪ Flooded surfaces ▪ Flood trace ▪ Volume balance ▪ Hazard map ▪ Representative values ▪ …

6	**Display Results** ▪ Long Profiles ▪ Time Series / Movies ▪ 2-D representation of topography, level differences, water depths, flow field, streamlines, vorticity, shear stress, etc. ▪ 3-D representation of topography ▪ Energy line ▪ Transport diagram ▪ Cross sections ▪ …

4.2 Scenario Examples

Depending on the chosen scenario and on the available boundary conditions or i.e. topography data, the approach for a successful simulation differs from case to case. As there are different ways to reach a certain target, the following activity diagrams just present possibilities but not a strict guideline.

An important part is always the grid generation. Usually, the raw topography data needs a lot of treatments (manual correction, interpolation, adjustment of single elements, etc.) until a suitable computational mesh can be generated. Although programs for grid generation like SMS provide some powerful tools to manipulate mesh transformations, the user still has to retain an overview over the required steps leading to the final computational mesh.

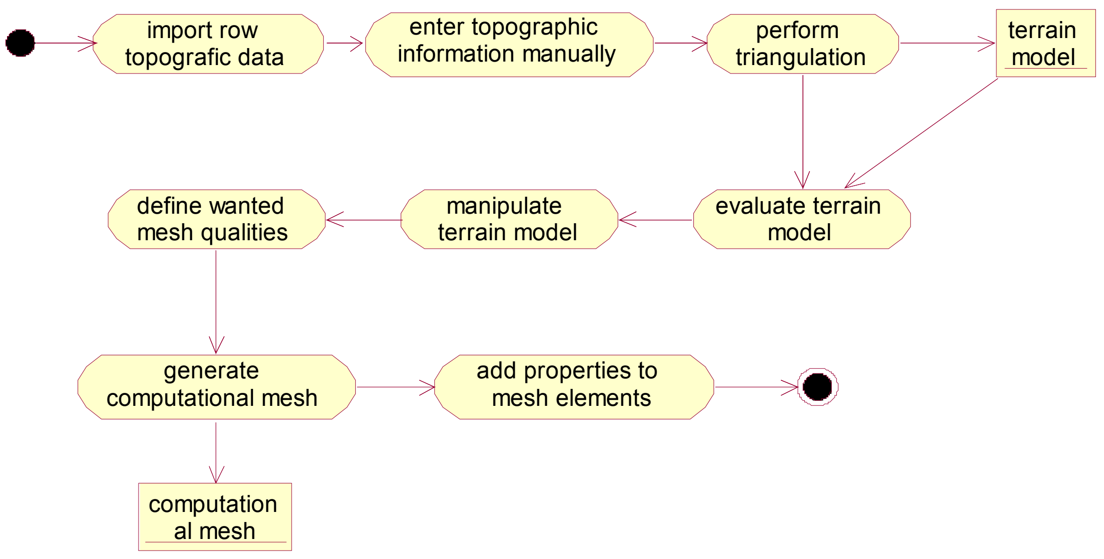

import row topografic data
enter topographic information manually
perform triangulation
terrain model
define wanted mesh qualities
manipulate terrain model
evaluate terrain model
generate computational mesh
add properties to mesh elements
computation al mesh

Fig. 6: Activity diagram generate computational mesh

The following activity diagrams show a possible procedure for different project scenarios.

4.2.1 Sediment balance in a river 1-D

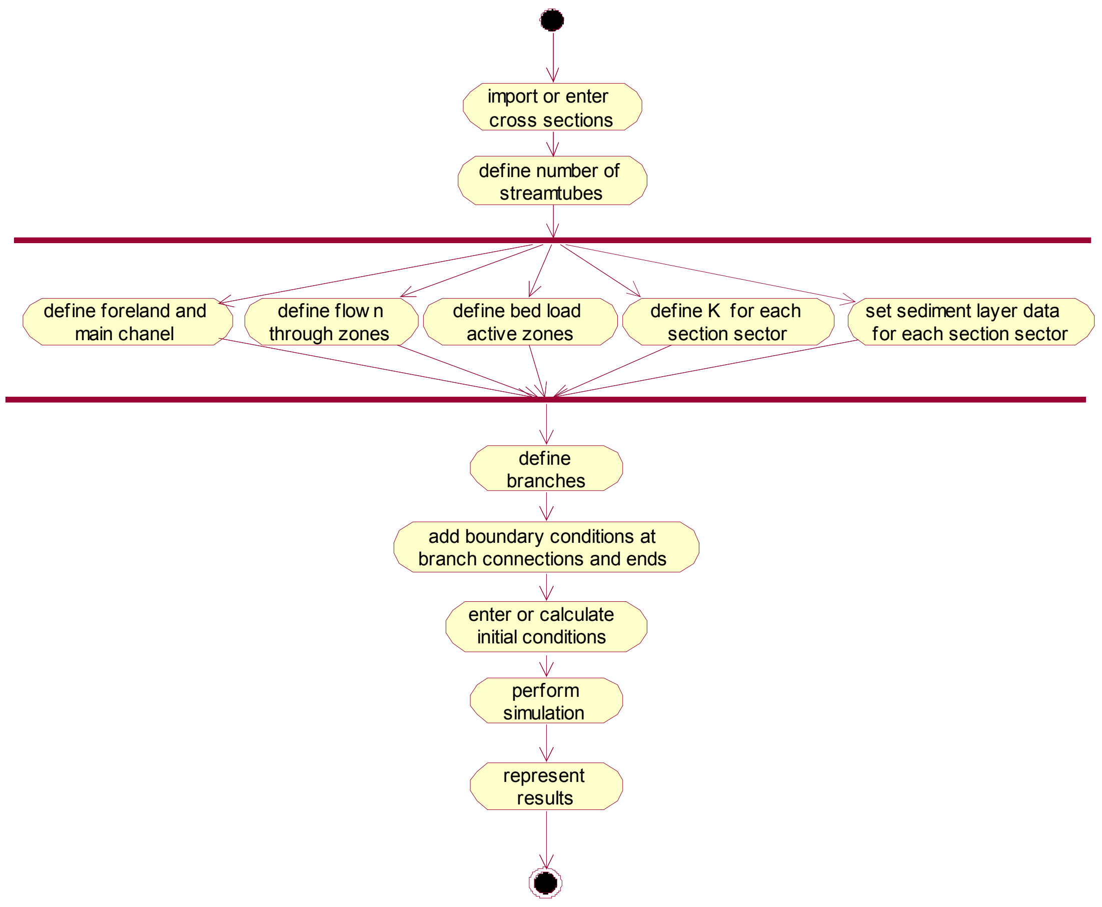

import or enter cross sections
define number of streamtubes
define foreland and main chanel
define flow n through zones
define bed load active zones
define K for each section sector
set sediment layer data for each section sector
define branches
add boundary conditions at branch connections and ends
enter or calculate initial conditions
perform simulation
represent results

Fig. 7: Activity diagram Sediment balance in a river

4.2.2 Flood 1-D + 2-D

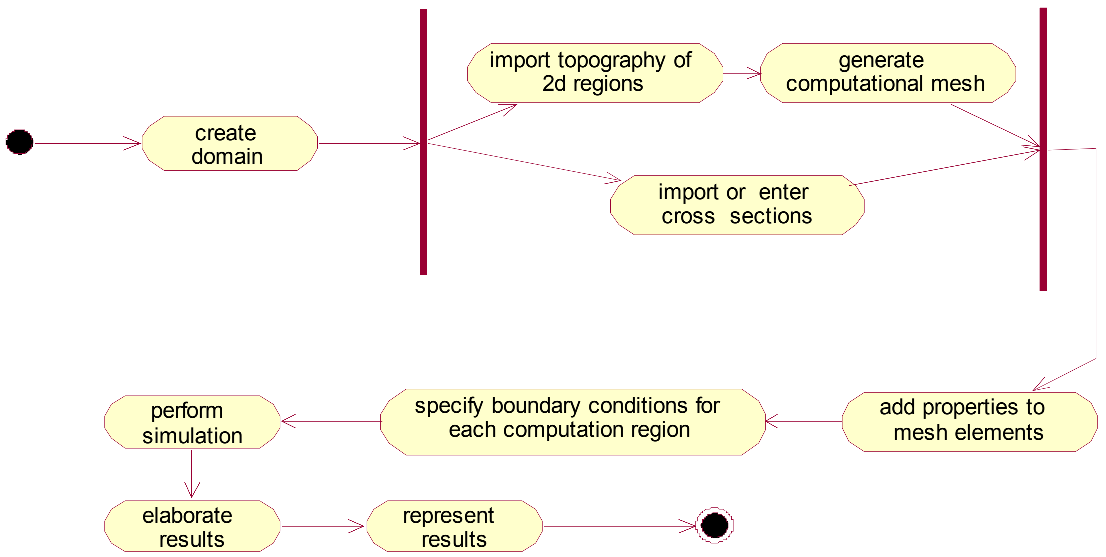

create domain
import topography of 2d regions
generate computational mesh
import or enter cross sections
add properties to mesh elements
specify boundary conditions for each computation region
perform simulation
elaborate results
represent results

Fig. 8: Activity diagram Flood 1-D + 2-D

4.2.3 Debris flow 2-D

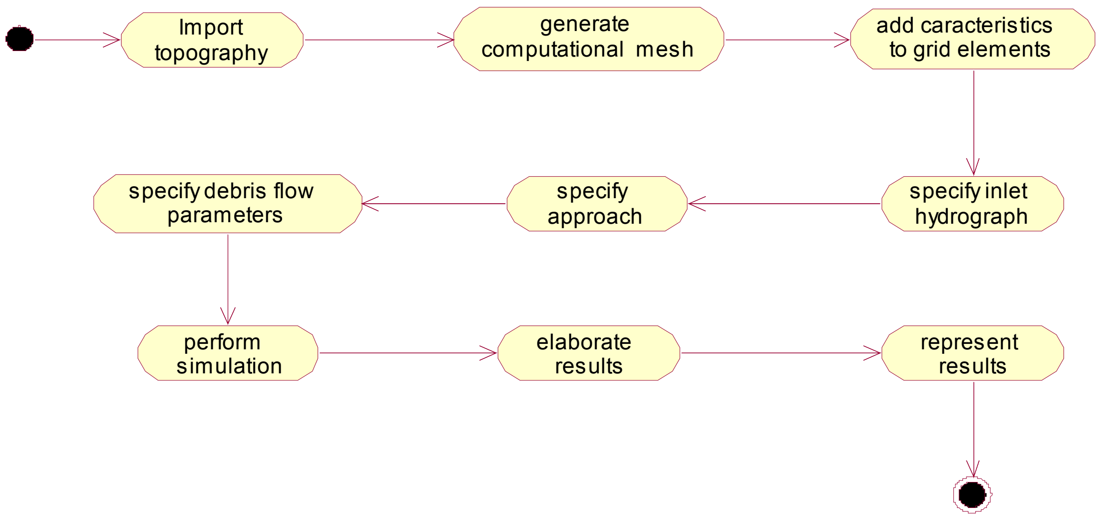


Fig. 9: Activity diagram Debris flow 2-D

This page has been intentionally left blank.

4.3 Executing BASEMENT

4.3.1 Executing BASEMENT with graphical user interface (GUI)

The start and executing of the BASEMENT software is described in the part "Introduction and Installation" of this manual. Further details concerning the GUI of BASEMENT are explained in the user manual UIII.

4.3.2 Executing BASEMENT on Microsoft Windows

When running BASEMENT under Microsoft Windows operating system, the easiest way to start a simulation is by double clicking on the command file ending with ".bmc".
Otherwise the program can be executed choosing the "Run…" command for the Windows "Start"-menu or by double–clicking the executable file in Windows Explorer. After running, BASEMENT will open the graphical user interface where the command file can be loaded and the simulation can be started.

BASEMENT creates an initialization file in the user's HOME-directory 'bm.ini', which stores the present work directory and scenario name to ease the input procedure for repeated simulations of the same scenario.

According to the existence of a main control block, either *BASECHAIN_1D* or *BASEPLANE_2D*, the appropriate simulation will be carried out.

4.3.3 Executing BASEMENT on Linux

BASEMENT runs as a console application without screen graphics output. On LINUX you open a console and type 'BASEMENT_vX.Y' (replace X.Y with current version number) to start the executable (if no environment variables have been set, change into your 'bin' directory of the installation path). After running, BASEMENT will open the graphical user interface where the command file can be loaded and the simulation can be started.

BASEMENT creates an initialization file (as a hidden file) in the user's HOME-directory '.bm.ini', which stores the present work directory and scenario name to ease the input procedure for repeated simulations of the same scenario.

According to the existence of a main control block, either BASECHAIN_1D or BASEPLANE_2D, the appropriate simulation will be carried out.

4.3.4 Executing BASEMENT in batch mode

Executing a simulation with BASEMENT normally opens the graphical user interface (GUI) and requires some input from the user, e.g. to select the model data and to confirm warnings generated by the program at the start and during run-time. But BASEMENT can optionally be started without any graphical interaction and without user input. This feature is especially useful if one or several models shall be run automatically via batch or script file.

But be aware that executing in batch mode requires special attention, since significant warnings may be suppressed without being noticed! It is recommended to study the generated 'log-file' after the simulation to check the program output for warnings which may have been generated during run time.

Executing in batch mode can be specified at the program start of BASEMENT using program arguments. The following list of program arguments is supported at the moment and can be specified in any order.

Command line arguments	*Description*
-b	BASEMENT is run in batch mode without manual user input.
-f filename	The file flag '-f' and the space separated 'filename' argument specify the model filename which shall be executed. The filename must be the full path including the name of the *.bmc-file. (Please note: No empty spaces are allowed to be part of the filename!)
-version	Display the version number of the BASEMENT executable.
-h	This help flag '-h' displays all available command line arguments.
-doc	This generates a reference documentation of all blocks and parameters in .html format.

Example:
Type the following line to execute the model 'Scenario1.bmc' in batch mode without user input.

```
BASEMENT_vX.X.exe -f d:\data\Scenario1.bmc -b
```

4.3.5 Restart the simulation in BASEplane from a given solution

In 2D simulations with BASEplane an improved and enhanced method for the restart from existing solutions from old simulation runs was implemented. Such a 'restart-file' contains all relevant information and data which is needed for the continuation of an old simulation and therefore often needs a lot of disk storage. These data are now stored in a binary format to reduce the needs for disk storage and to obtain smaller files. For this purpose a standardized CFD format was chosen (CGNS – CFD General Notation System, www.cgns.org). This standardized data format additionally can be edited and visualized by different programs and simplifies data exchange between different programs. (A simple Data Viewer for CGNS files is the *adfviewer* which can be found on www.cgns.org). The restart from an old solution is possible for the hydraulic computations, bed load transport computations and suspended load computations.

Using this new file format the possibilities to continue a simulation from a given solution were enhanced. It is possible to continue a simulation not only from the last point in time of the old simulation run, but also from different times of the old simulation. This can be very helpful especially for large simulation runs with long durations. For example, if there was an error in the inflow hydrograph at a point in time, than the simulation can be restarted shortly before the time when the error occurred with a corrected hydrograph and without having to repeat the whole simulation from beginning.

This page has been intentionally left blank.

5 Parallelization

5.1 Overview

5.1.1 Needs for parallel computing

Parallel computing is currently getting increasingly important for numeric scientific and engineering applications and this trend will become even more significant in the near future. Where parallel computers in the past were mainly limited affordable and manageable by large research institutions, today nearly all new available computers provide capacities for parallel computing. The famous law of Moore, which predicts an increase in computer power about a factor 2 every two years, is expected to be valid also in the near future due to parallel computing (Manferdelli 2008).

In parallel computing the increase of execution speed is mainly achieved by sharing the overall work load between several processors instead of further accelerating single processors by an increase of tact rates. Theses changes in computer hardware architectures raise the needs for new parallel programming concepts. Programs originally developed and optimized for execution on a single processor can not automatically benefit from the availability of multiple processors. Specific software techniques and algorithms must be developed and applied to exploit the additional performance provided by parallel hardware.
The software BASEMENT therefore needed to be adapted and optimized for the use of parallel computers.

Some typical hydrodynamic simulation scenarios with high demands on performance are

- multidimensional flow simulations with large computational domains consisting of a very large number of elements, as well as simulations of large river networks,
- simulations of river flow over long time periods (especially with regard to morphological changes) and
- real-time simulations of river flow and flood wave propagation.

5.1.2 Parallel computer architectures

Types of parallel computing can be differentiated in many aspects, concerning hardware, software concepts and various other criteria. Frequently, parallel computers are classified in two groups, which differ in the way memory is organized between the multiple processors.

In so called “distributed memory” architectures each core possesses its own memory unit, which can not be accessed by any of the other cores. Distributed memory systems usually consist of large computer clusters which are connected via a network (right side of *Fig. 10*). Such a parallelization concept enables the use of very large numbers of processors and is employed in high performance computing (HPC). But software programming, maintenance and debugging for distributed memory systems generally is very time-consuming and costly. Complex message passing interfaces via network are necessary for data exchange and synchronizations between the processors. Out of these reasons distributed memory systems, or combinations of distributed and shared memory systems, so called hybrid systems, are not further discussed here.

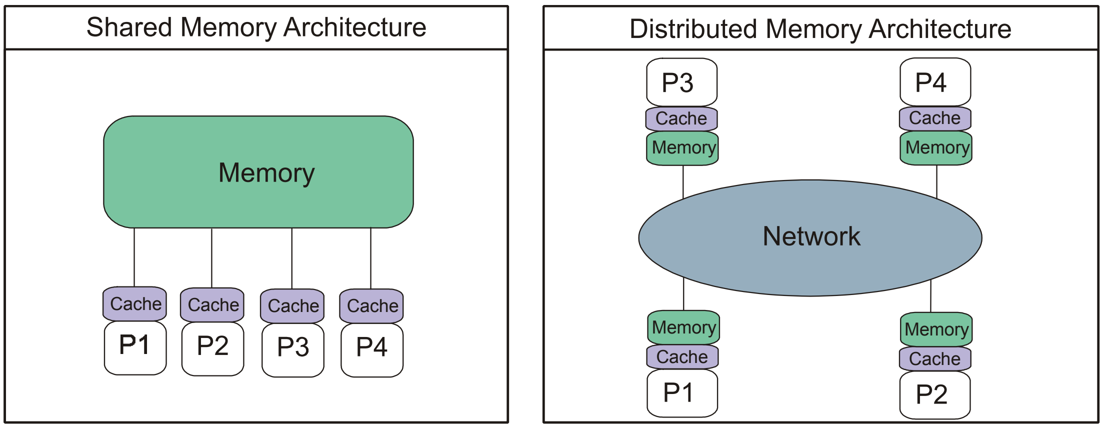


Fig. 10: Shared memory architectures (left) and distributed memory architectures (right) with four cores

On the other hand, “shared memory” architectures have only one single memory unit which is shared by multiple cores (*Fig. 10*, left side). The parallelization of the software BASEMENT is implemented for such shared memory architectures. This concept enables fast and easy communication between all cores through global access to memory. Furthermore, making benefit of parallel computing often is significantly easier and less time-consuming, especially for already existing sequential applications.
Such parallel computers with shared memory, often termed multi-core machines, are highly available today at low costs. The general trend in processor development is from multi core to many core systems with up to even more than hundred cores in cc-NUMA systems. As a consequence, a major drawback of shared memory parallelization, namely not being portable to distributed memory systems, seems acceptable for an application like BASEMENT which is not focused on high performance computing.

5.2 Parallelization issues on shared memory systems

5.2.1 Levels of parallelization

Parallelization in shared memory systems is based on multi-threading concepts. Threads are a way for a program to divide its work load into several independently running parts which are finally mapped on the available cores. Threads can be started ("forked") and terminated ("joined") by an application in order to exploit parallelism provided by multiple cores.

The distribution of work load on a series of threads can be made on a small scale loop level ("fine grained parallelism") or on a larger scale procedural or module level ("coarse grained parallelism"). See *Fig. 11* for illustration of these different concepts. Parallelizing on each of these levels has certain advantages and disadvantages. At the moment BASEMENT mainly implements a fine grained parallelism concept. In principle, both concepts can also be combined (nested parallelism).

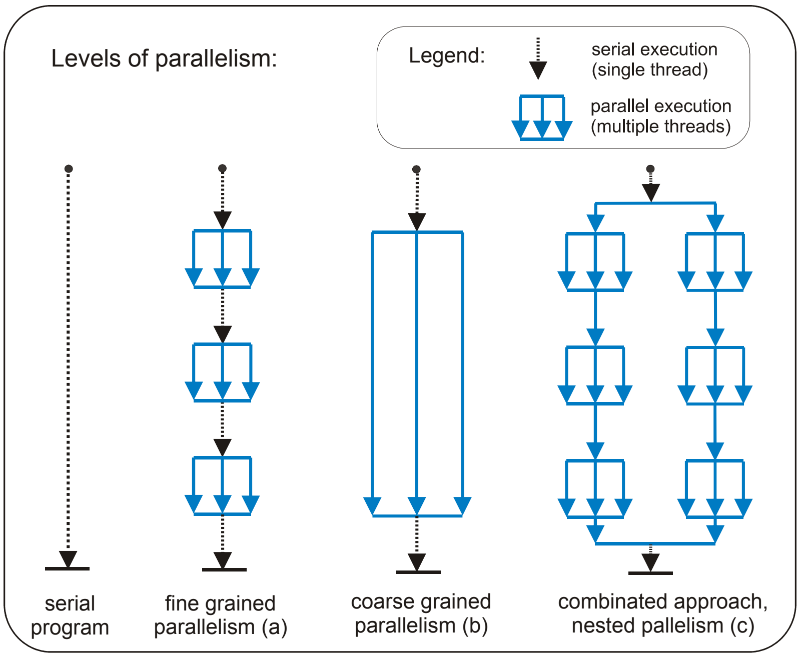

Levels of parallelism:
Legend:
serial execution (single thread)
parallel execution (multiple threads)
serial program
fine grained parallelism (a)
coarse grained parallelism (b)
combinated approach, nested pallelism (c)

Fig. 11: Iillustration of different levels of parallelism

Numerical applications like BASEMENT spend most of their execution time in rather small parts of the code. These parts are the main calculation loops which iterate over all elements and edges of the computational grid. Fine grained or loop level parallelization aims to exploit parallelism by parallelizing these time consuming loops. If the serial running program arrives at such a loop, its iterations are divided among a certain number of threads and are executed in parallel. Afterwards, a synchronization of the threads is performed so that the program can continue its serial execution (see *Fig. 11*). This parallelization approach usually requires no crucial changes to the source code and is especially suitable for parallelizing already existing serial programs. All time consuming loops can be parallelized step by step, thereby incrementally increasing the parallel performance of the program. Due to the small changes in the source code the robustness of a tested serial program is maintained.

Whereas most loops can easily be parallelized, some loops contain flow dependences where the calculation results depend on an ordered succession of all iterations, which is generally not maintained during parallel execution. Such flow dependences can often be resolved by reorganising some parts of the loop. Details on locating and removing flow dependences in loops can be found in literature (e.g. Chandra).

5.2.2 Factors influencing parallel performance

Parallel performance is often measured as speedup S(n), which is defined as the ratio between “serial execution time” and “parallel execution time” and states how much faster the parallel application runs compared to the serial one.

The theoretical maximum achievable speedup is limited to the number of cores n. The actual speedup is determined by several factors whose influences largely depend on size and type of the simulation. Furthermore the scalability indicates wether the speedup increases linearly with increasing numbers of processors or at a slower rate. The key factors influencing the speedup and according parallel programming aspects are briefly discussed in the following section.

- Parallel overhead

  Parallel overhead is created at many locations and times in the program. Threads must be forked and joined, loop iterations must be divided among threads and threads must be synchronized. In the worst case, the parallel overhead can even exceed the performance gains from parallel execution. This is especially a problem in case of small loops with small workload, e.g. in case of initialisation loops. Such loops are not suited for parallelization and should better be executed in serial or, if possible, should be integrated into larger loops.
  For simulations with loops of rather small workload but very high number of time steps, it is important to prevent parallel overhead from frequently joining and forking threads. This was achieved in Basement by integrating the whole time loop into a single parallel region.

  The significance of parallel overhead in a parallel application depends largely on the computational costs of the simulation domain. Simulations with high computational costs, e.g. simulations with large numbers of elements, are less negatively affected.

- <u>Coverage</u>

  In order to obtain a good parallel scalability it is important to parallelize large portions of the code resulting in a so called high “coverage”. A high coverage is of importance, because the negative impacts of remaining serial parts in the code increase as more and more cores are employed. The achievable speedup finally becomes limited by these serial parts (see “Amdahl’s law” (Chandra)).

- <u>Load balance</u>

  The overall execution time of a parallel loop is determined by the thread with the slowest execution time. If the work load is not properly balanced between the multiple threads performance will suffer. A good approach for parallelizing a loop often is a simple static scheduling, which means that all iterations are divided in parts of equal size before the threads start execution. But in case that the computational costs of the iterations differ largely, a static scheduling may not be optimal. E.g., in 2-D simulations with dry and wet regions, the flow equations must only be solved entirely for the wet or transitional elements but not for the dry elements.
  Beside the problem of moving boundaries of the wetted domains, it is also important to note that hydrodynamic models may lead to imbalanced load distributions in case of local, highly unsteady processes like wave propagations.
  The load balance can sometimes be optimized in such cases by the use of dynamic scheduling, which divides the work load dynamically among the threads. Threads with computational cheap iterations will dynamically receive additional iterations, thereby taking load from threads with costly iterations. Dynamic scheduling can improve performance in case of load imbalance, but leads also to an increased overhead and suffers from bad data locality. Consequently, the decision between static or dynamic balancing is largely problem dependent.

- <u>Synchronization</u>

  Synchronization between threads can become time consuming if some threads must wait for other threads to finish execution. Quite similar, in some cases threads must be prevented from mutual accessing the same memory locations to prevent read/write conflicts and data races. Such parts of the code must be protected by setting mutual exclusions using locks. These code parts can only be accessed by one thread a time and may cause other threads having to wait until access is granted.
  Such bottlenecks caused by synchronisations and mutual exclusions should be avoided and minimized as far as possible. They can sometimes be prevented by reorganising parts of the algorithms. Prevention of memory conflicts can sometimes also be achieved by privatization of the problematic shared variables, i.e. global variables are replaced by thread private variables.

- Locality

  When parallelizing a program attention should be paid on data locality aspects. Modern cache based processors optimize execution speed by caching data which speeds up data load and store cycles. Caches are very efficient if a processor can use the cached data (the local data) for many operations instead of having to reload them from memory or from caches of other processors. In general, better data locality is guaranteed if the work load is distributed statically among the threads instead of using a dynamic schedule.

  Additional problems limiting the parallel performance may arise when OpenMP parallelized programs are executed on cc-NUMA architectures with large number of cores due to general aspects of the underlying hardware architecture (Chapman 2008).

5.3 Parallelization with OpenMP

5.3.1 Overview

As mentioned before, parallel programming for shared memory systems bases on multi-threading software concepts. Thread programming can be done on a rather complex and time consuming low level by explicitly specifying and controlling all threading options. In recent years high level concepts for thread based parallelization were developed which ease parallelization and do not require low level thread programming skills. Among these high level concepts it can be differentiated between implicit parallelism using parallel programming languages or automatic parallelization by compilers and explicit parallelism where the developer controls the parallelism with support of parallel libraries and application programmer interfaces (APIs).
In cooperation with leading software and computer enterprises a parallel API called OpenMP ("Open Multi Processing") was developed which is today the de facto standard for parallelizing scientific and engineering applications on shared memory systems. OpenMP is used for the parallelization of BASEMENT.

5.3.2 Characteristics of OpenMP

OpenMP consists of a set of compiler directives and library functions. With these compiler directives the developer describes the parallelism in the code. Therefore a compiler is needed which supports these OpenMP directives. OpenMP is currently supported by many C/C++ and Fortran compilers on a variety of platforms. In the table below some aspects pro and contra parallelization with OpenMP are listed (see also Kuhn for a comparison of OpenMP and threading in C/C++).

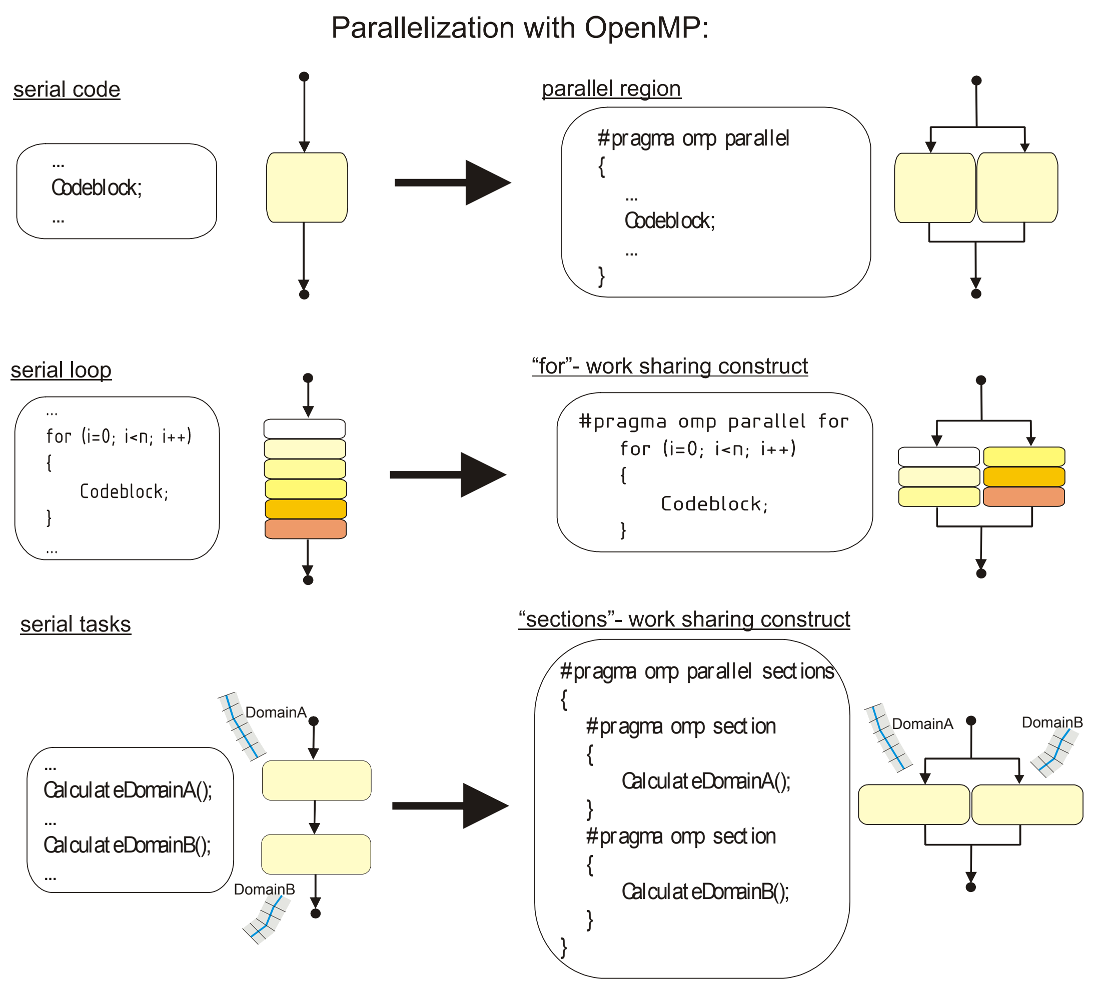
Pro	Contra
First successes in parallelization are relatively easy to achieve. Compiler directives reduce the needed programming efforts and implementation details are left to the compiler.	No full control over the implementation details is possible. Bugs in libraries can be difficult to isolate.
Rather small increases in the size of the source code due to the use of compiler directives. Good readability of the code is maintained and the algorithms are not buried by large blocks of added parallelization instructions.	Compiler generated code portions and library calls can complicate debugging of the parallel program.
OpenMP is standardized and portable on different platforms.	OpenMP does not support the whole range of threading concepts (e.g. no support for semaphores for synchronization).

The source code can be compiled as serial program without support of OpenMP (compiler directives are treated as comments). This may ease debugging of new implemented application features.	

Tab. 1: OpenMP parallelization –pro and contra

5.3.3 Parallel directives in OpenMP

The basic parallelization construct in OpenMP is the so called "parallel region". The source code placed within a parallel region is executed in parallel on a number of threads. At the end of a parallel region an implicit barrier is automatically set to synchronize the threads.


Parallelization with OpenMP:
serial code
parallel region
serial loop
"for"- work sharing construct
serial tasks
"sections"- work sharing construct
DomainA
DomainB
DomainA
DomainB

```
...
Codeblock;
...
```

```
#pragma omp parallel
{
    ...
    Codeblock;
    ...
}
```

```
...
for (i=0; i<n; i++)
{
    Codeblock;
}
...
```

```
#pragma omp parallel for
    for (i=0; i<n; i++)
    {
        Codeblock;
    }
```

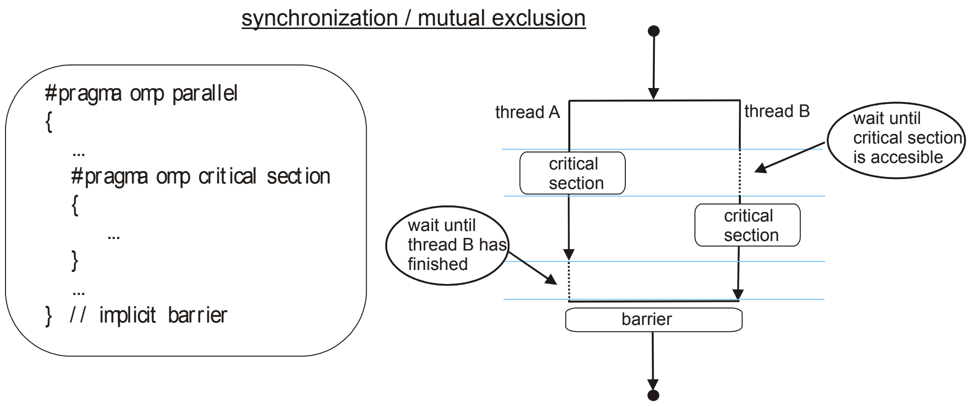
```
...
CalculateDomainA();
...
CalculateDomainB();
...
```

```
#pragma omp parallel sections
{
    #pragma omp section
    {
        CalculateDomainA();
    }
    #pragma omp section
    {
        CalculateDomainB();
    }
}
```

Fig. 12: Compiler directives for basic parallelization constructs in OpenMP

To ease the distribution of work load among the threads of a parallel region, OpenMP supports work sharing constructs which automate such tasks, see *Fig. 12*. With optional parameter clauses it is possible for the user to define the behaviour of these compiler directives in more detail. A complete OpenMP reference is available online at (http://www.openmp.org). For special needs it is also possible to share the work load among the threads fully flexible by individually addressing each thread.

In OpenMP several compiler directives are supported for synchronization issues which cover the most frequent tasks. Such constructs are barriers, locks, critical sections and atomic constructs. Beside these standard constructs, fully flexible synchronisation operations can be implemented for special tasks using global flags in memory. The choice of the best suited constructs for synchronisation and mutual exclusion is problem depended and involves differing efforts in parallel programming.


synchronization / mutual exclusion
thread A
thread B
wait until critical section is accesible
critical section
critical section
wait until thread B has finished
barrier

```
#pragma omp parallel
{
   ...
   #pragma omp critical section
   {
      ...
   }
   ...
} // implicit barrier
```

Fig. 13: Critical sections and barriers for synchronization operations in OpenMP

Some general parameters controlling the parallelism can be set in OpenMP at the program start or during runtime with library calls or by setting environment variables. Such parameters include, for example, the number of parallel threads or the choice between static or dynamic scheduling. This enables the flexibility to let the user determine the number of threads used for parallelization or to optimize the parallel speedup by varying the type of schedule.

This page has been intentionally left blank.

6 Model Coupling

6.1 Introduction

In addition to the simulation of single sub-domains using BASEchain (1-D) or BASEplane (2-D), the software BASEMENT also provides the possibility to connect sub-domains for combined numerical simulations. Such coupled simulations can range from simple configurations up to simulations of river networks with integrated river junctions / bifurcations or integrated 1-D/2-D modelling. In Fig. 14 a river network of multiple sub-domains with several coupling interfaces is illustrated. The coupling mechanisms thereby allow to couple hydrodynamic simulations as well as morphological simulations with sediment transport and suspended load.

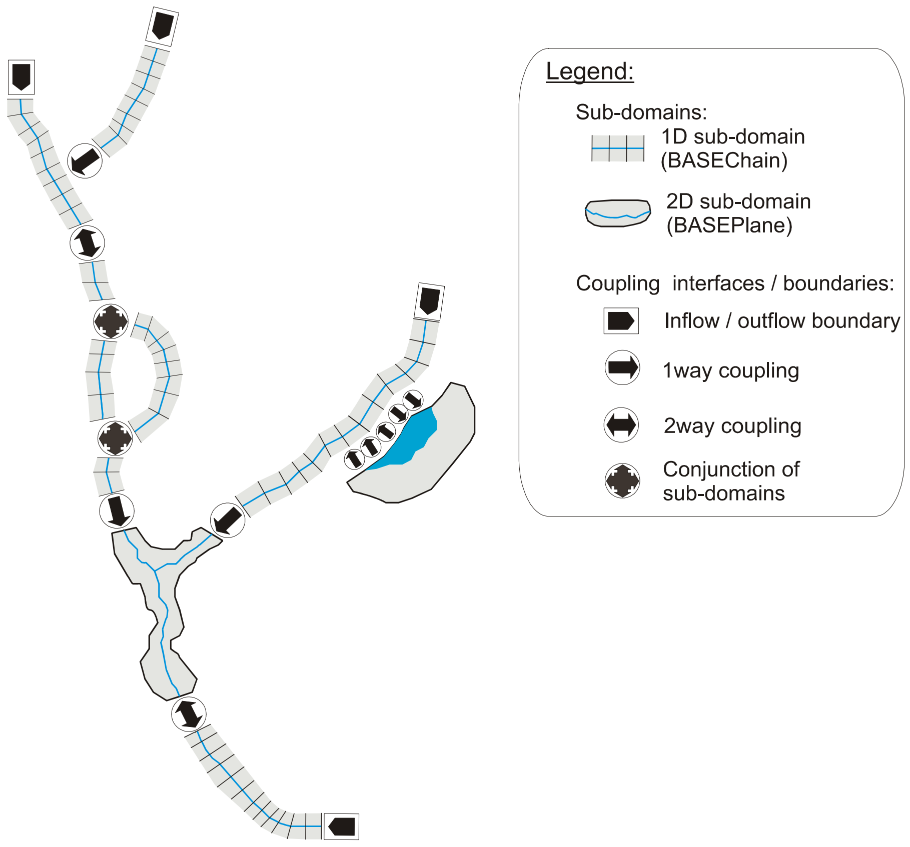

Legend:
Sub-domains:
1D sub-domain (BASEChain)
2D sub-domain (BASEPlane)
Coupling interfaces / boundaries:
Inflow / outflow boundary
1way coupling
2way coupling
Conjunction of sub-domains

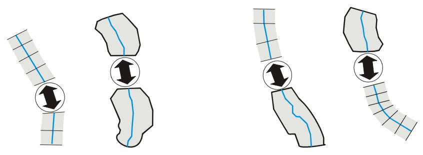
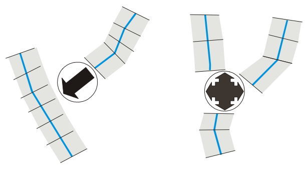
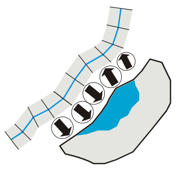
Fig. 14: River network with multiple BASEchain (1-D) and BASEplane (2-D) sub-domains and several coupling interfaces.

Some typical applications of coupled simulations are:

- A step wise modeling approach to the overall problem using smaller parts of the whole domain. This approach has the advantages of reduced complexity and reduced execution and calibration times. Also extensions of existing and calibrated models can be easily made with coupled simulations without the need to redesign the existing models.
- Simulations with hydraulic structures (like weirs or gates) within the domain of interest can be realized by using multiple sub-domains, which are coupled via these hydraulic structures.
- Coupled simulations can be helpful for mixed-dimensional modeling approaches, e.g. for cases where large scale 1-D simulations shall be combined with detailed modeling of local areas in 2-D. Thereby, the advantage of efficient and robust modeling in 1-D is combined with the capability to simulate 2-D flow characteristics. Also, the required efforts for data acquisition and data preparation can be minimized using mixed-dimensional modeling approaches.

6.2 Coupling Types

The implemented coupling types are briefly sketched in the table below.

Coupling types	*Description*
 *1-D / 1-D, 2-D / 2-D* *1-D / 2-D 2-D / 1-D*	Single sub-domains can be combined sequentially via coupling interfaces at the upstream or downstream boundaries. This can also be done for sub-domains with mixed dimensionalities (1-D / 2-D, 2-D / 1-D). These sequential coupling types can be used to combine sub-domains over their boundary conditions or external sources. For example, a weir outflow boundary can be combined with an input hydrograph of a downstream boundary.
 *junctions / bifurcations*	Coupling interfaces for river junctions or river bifurcations allow a simplified modelling of conjunctions of river branches within a 1-D river network.
 *lateral coupling*	For integrated 1-D and 2-D modelling, a 1-D sub-domain can be coupled laterally with a 2-D sub-domain. The coupling takes place along the river channel via multiple coupling interfaces which connect cross sections (1-D) with corresponding mesh elements (2-D).

Tab. 2: Listing of different coupling types and their descriptions.

This page has been intentionally left blank.

6.3 Coupling Mechanisms

Explicit coupling of sub-domains

Coupling of sub-domains is implemented as an explicit coupling approach, which means that data is exchanged explicitly between the sub-domains at certain time intervals. This approach is simpler to implement than an implicit approach, especially regarding the coupling of sub-domains with mixed dimensionalities. However, in comparison to an implicit coupling approach, special care must be taken to achieve robust and stable combined simulations.

6.3.1 One-way coupling and two-way coupling

A simple way to couple two sub-domains is to exchange data only in one direction from upstream to downstream. Such a situation is termed as 1-way coupling from here on. It has the advantage that the upstream sub-domain can run independently from the downstream sub-domain and the flow variables are passed over at some time intervals to the downstream sub-domain. But being a one-directional coupling, no information from downstream can travel upstream. Therefore, this type of coupling is restricted to cases where no backwater effects from downstream take place or such influences can be neglected.

In contrast, a two-way coupling enables mutual interactions between the sub-domains by providing mutual data exchange. In two-way coupled sub-domains, backwater effects from downstream can influence the upstream sub-domain. Instead of executing the sub-domains sequentially from upstream to downstream direction, here the sub-domains are executed simultaneously.
The two-way coupling approach has the difficulty that no unique flow variables are present at the coupling cross sections, as water levels from upstream and downstream direction may differ for a given time. In principle, iterations between the sub-domains are required and must be performed until the differences of the variables at the coupling cross section do no longer change within subsequent iteration steps. Although a rather small number of iterations has to be expected (as reported by Miglio, Peretto and Saleri (2005)), these iterations lead to large additional computational efforts. Therefore these iterations are not performed here. As the time steps in the explicit approach are usually very small, the differences between the upstream and downstream variables are rather small and iterations may be neglected without substantial loss of accuracy. But in cases of crucial and abrupt changes in the flow variables, oscillations may result.

A combination of both concepts can be used in the coupled river simulation. one-way coupled sub-domains are executed sequentially from upstream to downstream direction, whereas two-way coupled sub-domains are treated as being a single sub-domain within the execution sequence

This page has been intentionally left blank.

6.4 Definitions of Exchange Conditions

6.4.1 General remarks

Data can be exchanged between the sub-domains by coupling interfaces using boundary conditions and source terms. The following table shows the exchange variables grouped by the direction of the exchange.

direction of exchange	*type of coupling*	*exchange variables*		
in downstream direction	boundary conditions & sources	Q discharge	$q_{b,g}$ bed load	C_g Concen-tration
in upstream direction	boundary conditions & sources	z_S water surface elevation		

Tab. 3: Possible exchange conditions between sub-domains.

In order to enable simple, flexible and efficient coupled simulations, some assumptions are made here:

- It is assumed that flow directions at the coupling interfaces of the river network are known a priori and do not change during the simulation (with the exception of special coupling types, like the *lateral coupling*).
- The cross sections (1-D) or mesh elements (2-D) of the coupling interfaces should ideally be located at the same or nearby locations and have the same geometries. This is necessary to reduce possible errors around the coupling interfaces due to the disregard flow taking place in between and to avoid discontinuities due to abrupt changes in the geometries.
- It is assumed that the flow is orthogonal over the boundaries, i.e. the directional x- and y flow components in 2-D are not exchanged separately.
- In 2-D coupling only summarized or averaged data are exchanged, instead of exchanging data separately for each edge or element. This approach simplifies the coupling setup since no restrictions are set regarding the geometries and number of cells at the boundaries or sources.

6.4.2 Exchange conditions for mixed-dimensional sub-domains

Exchange via boundary/sources:

Exchange variable	*Exchange equations*	*Nr of exchange terms*
Discharge	1-D→2-D: $q_i^{2D} = \omega_i Q^{1D}$ (ω_i = area/length weighting or conveyance weighting) 2-D→1-D: $Q^{1D} = \sum_i^n q_i^{2D}$	1
Water surface	1-D→2-D: $z_{S,i}^{2D} = z_S^{1D}$ 2-D→1-D: $z_S^{1D} = \frac{1}{n}\sum_i^n z_{S,i}^{2D}$	1
Bed load	1-D→2-D: $q_{b,g,i}^{2D} = \omega_i q_{b,g}^{1D}$ (ω_i = area/length weighting) 2-D→1-D: $q_{b,g}^{1D} = \sum_i^n q_{b,g,i}^{2D}$	$g = 1..ng$
Suspended load	Not available yet in 2-D	$g = 1..ng$

Tab. 4: Exchange conditions for mixed-dimensional coupling (i = index of 2-D edge or 2-D element)

6.4.3 Exchange conditions for river junctions in 1-D river networks

Within a river network locations are encountered where river branches flow together or where a river bifurcates into several branches. The flow characteristics at such conjunctions generally are multidimensional. Therefore the preferable modelling approach to achieve a good accuracy is to simulate a 2-D sub-domain. But if such a situation shall be modelled with 1-D sub-domains than special coupling concepts are required.
Two different approaches are implemented in BASEMENT (see Fig. 15). These approaches allow no more than three sub-domains being part of a junction. If a larger numbers of river branches are to be modelled, they must be approximated by multiple junctions, placed in small distances.

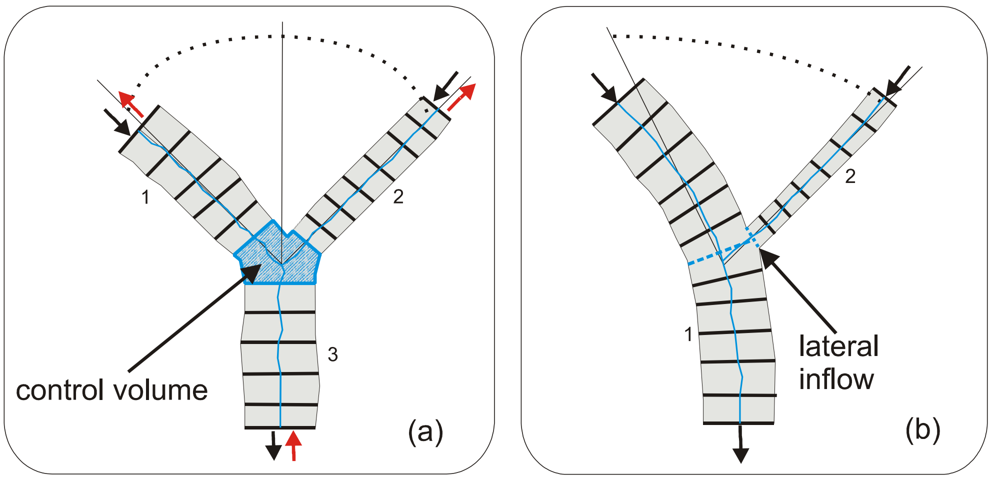


Fig. 15: Modeling of a river junction with two different approaches (black arrows indicate a confluence of river branches, red arrows a bifurcation).

Following the first approach a junction can be regarded as region where three different river branches meet and mutually exchange data (a). A control volume is defined to which mass and momentum conservation principles can be applied. A simple approach is here balancing discharges and assuming equal water surface elevations along the junction.

	Exchange conditions	*Nr. of equations*
Discharge	$Q_{up1} + Q_{up2} = Q_{down}$	1
Bed Load	$Q_{up1\_bed,g} + Q_{up2\_bed,g} = Q_{down\_bed,g}$	$g = 1..ng$
Suspension	$Q_{up1}C_{up1,g} + Q_{up2}C_{up2,g} = Q_{down}C_{down,g}$	$g = 1..ng$

Tab. 5: Exchange conditions for river junctions

The second approach is to regard the junction as a lateral inflow of a tributary into a river at a specified location (b). The discharge (and sediment) is passed from the tributary to the river as lateral inflow via source term. Additionally, the water level at the inflow cross section can be passed in return to the tributary. Despite its simplicity this approach can be suited well to simulate simple river junctions in 1-D.

6.4.4 Exchange conditions for river bifurcations in 1-D river networks

In case of modeling a river branch which bifurcates into two branches, the upstream discharge (and sediment) must be distributed among the two downstream sub-domains. The distribution factor ϕ among the downstream sub-domains has to be chosen according the local conditions. The downstream water elevations of the two downstream sub-domains are averaged and then passed in upstream direction.

	Exchange conditions	*Nr. of equations*
Discharge	$Q_{up} = \phi Q_{down1} + (1-\phi)Q_{down2}$	1
Bed Load	$Q_{up\_b,g} = \phi Q_{down1\_b,g} + (1-\phi)Q_{down2\_b,g}$	$g = 1..n$
Suspension	$Q_{up}C_g = \phi Q_{down1}C_{down1,g} + (1-\phi)Q_{down2,g}C_{down2,g}$	$g = 1..n$

Tab. 6: Exchange conditions for river bifurcations

6.4.5 Exchange conditions for combined 1-D and 2-D modelling

The combined 1-D river flow and 2-D floodplain modelling bases mainly on the approach presented by Beffa (2002). A conceptual overview is given in Fig. 16 which illustrates river cross sections of the BASEchain sub-domain and the 2-D mesh of a floodplain modelled with a BASEplane sub-domain.

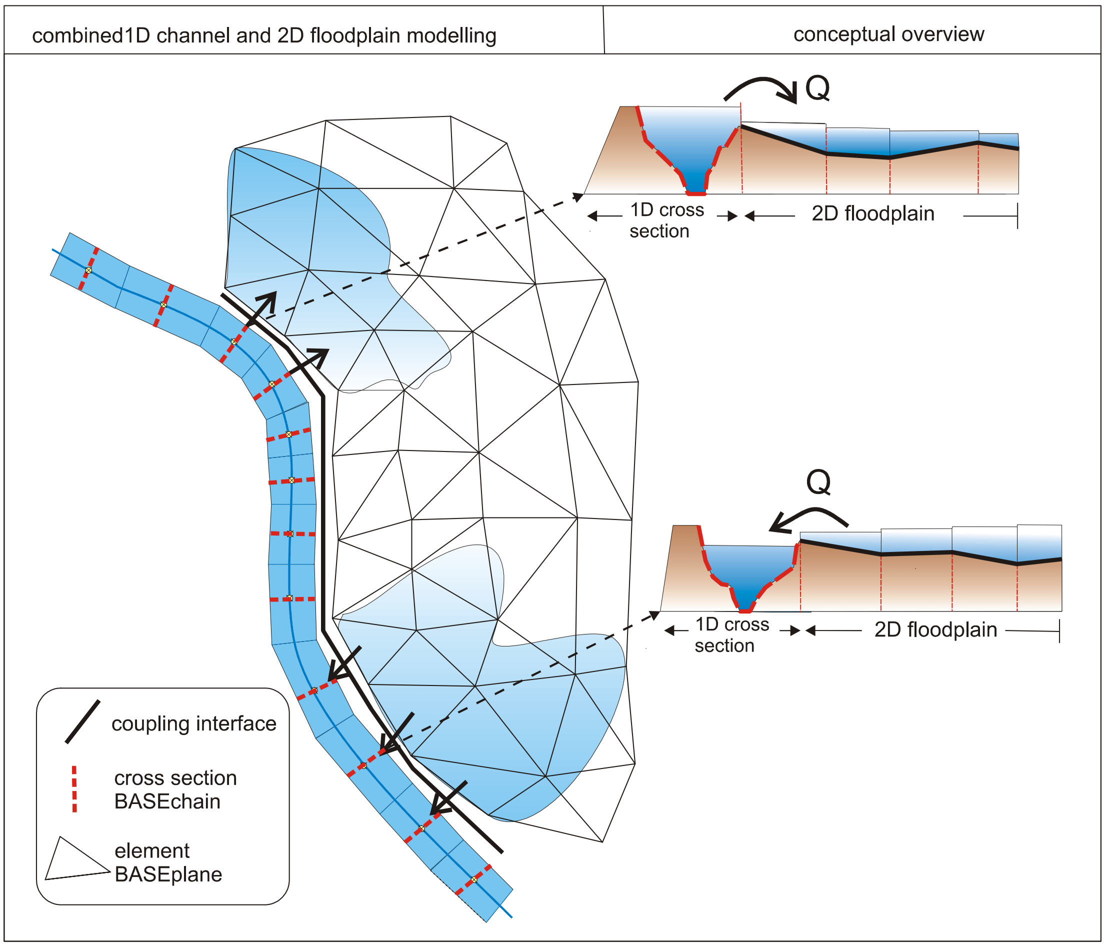


Fig. 16: Conceptual overview of combined 1-D river flow and 2-D floodplain modeling

The coupling interfaces between the sub-domains are implemented as one-way couplings via source terms. As a consequence, only discharges are exchanged between the sub-domains. The exchange between the sub-domains is calculated as weir flow over the dykes of the 1-D cross section or as weir flow over the edges of 2-D sub-domain. The weir level is chosen as the higher elevation of the dyke or the corresponding edge. As weir width b in the weir formula the length of the 2-D boundary edge is taken. Exchange of discharge is possible in both directions, either from the river into the floodplains or backwards depending on the water elevations in the 1-D cross section and the corresponding 2-D element.

To enable a flexible coupling approach it is possible to connect a 1-D cross section with multiple 2-D elements. The coupling interfaces are defined using a list, from which the connections are automatically extracted and generated during a pre-processing step.

	Exchange conditions	*Nr. of equations*
Discharge	1-D→2-D: $Q = \delta\mu\frac{2}{3}b_{weir}\sqrt{2g}h^{\frac{3}{2}}$ if $(z_{S,1D} \geq z_{S.2D})$ (side weir, δ = side weir reduction factor) 2-D→1-D: $Q = \mu\frac{2}{3}b_{weir}\sqrt{2g}h^{\frac{3}{2}}$ if $(z_{S,1D} < z_{S,2D})$ (weir overfall over edge)	1

Tab. 7: Exchange conditions for lateral coupling

6.4.6 Data exchange for morphological simulations with multiple grain classes

In morphological, coupled simulations the possibility exists that sub-domains can have differing grain compositions. The handling of data exchange for such cases is not trivial and unclear. But such situations may arise in coupled large-scale simulations where grain classes get finer along course of the river. A flexible approach is adopted here which allows the usage of differing compositions as well as different numbers of grain classes of the sub-domains.

For data exchange the bed loads of each grain class are mapped on the grain classes of the receiving sub-domain. The mapping is achieved by three successive steps as illustrated in Fig. 17. The sediment mass balance is thereby fulfilled.

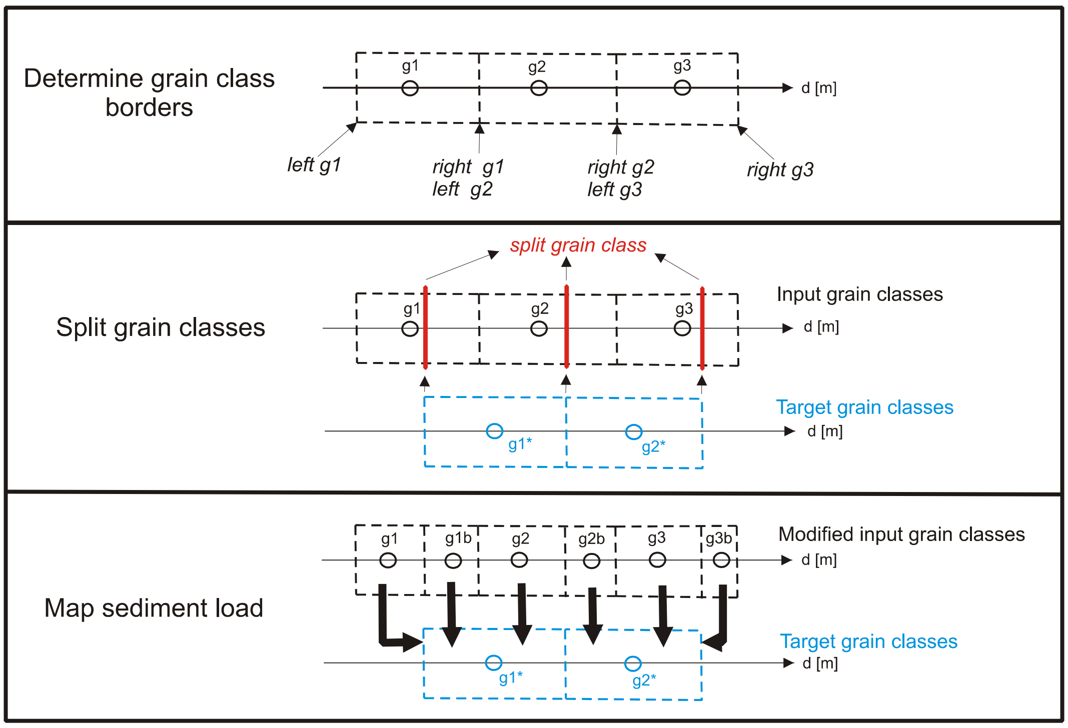


Fig. 17: Mapping of grain compositions from one sub-domain to another

6.5 Synchronization Concept

6.5.1 General remarks on synchronization

For the coupling of sub-domains a synchronization mechanism must be implemented which directs the execution of the sub-domains and controls the data exchanges at the appropriate times. The type and complexity of the synchronization effort thereby generally depends on the degree of spatial and temporal compatibility of the sub-domains. Especially in case of combined 1-D and 2-D simulations the spatial extends and time step sizes can vary considerable.

Mainly two different coupling concepts are often encountered for the selection of the time step sizes of the sub-domains.

- All sub-domains are executed in a synchronous manner with an equal time step size. To guarantee stable execution the chosen time step size (“global time step”) is set to the minimum time step size of all sub-domains, which is determined by stability conditions (CFL criterion). But due to the fact that the sub-domains’ time step sizes can vary considerable, such a restriction on the minimum time step size can lead to inefficient small time step sizes resulting in large computational efforts.
- In contrast, all sub-domains can be executed asynchronous with different time step sizes (“local time steps”), which are chosen according the sub-domains´ optimal time step size. This approach does not suffer the computational inefficiency due to small time step sizes. But generally more synchronization efforts a required and data exchange between the sub-domains requires interpolations and can become cumbersome especially for complex interfaces like junctions or bifurcations.

Here, another approach is selected, a local-time stepping approach, lying in between these concepts and combining efficiency and simplicity.

6.5.2 “Local time stepping” approach

This approach bases on the method of local time stepping (LTS) as presented by Osher and Sanders (1983) and Sanders (2008). But in contrast to these methods, LTS is applied here to whole sub-domains instead of single grid elements. Different local time step sizes are allowed for the sub-domains instead of using one global time step for all sub-domains. This enables efficient computations by preventing very small time steps of single sub-domains to dominate the time step sizes of the other sub-domains. But restrictions are set for the time step sizes in a way to ensure that the sub-domains always reach common time levels. At these common time levels data can be exchanged easily without the need for interpolations.

Hierarchical levels L are introduced and attributed to each sub-domain. These levels categorize the sub-domains into groups of common time step sizes. These levels are thereby chosen as power-of-two multiples of the base time step size Δt_{base}. This base time step is selected as the minimum time step size of all coupled sub-domains. The attribution of levels L to a sub-domain i depends on the relation of its present time step size to the base time step size and is determined as:

$$2^k \le \frac{\Delta t_i}{\Delta t_{base}} < 2^{k+1} \quad \Rightarrow \quad L_i = 2^k, \ \ k = 0..n$$

where L_i is the level attributed to the sub-domain i, k indicates the level and n is the number of levels. Each sub-domain determines its own local time step size as its level L_i multiplied with the base time step size $\Delta t_i = L_i \Delta t_{base}$.

The execution of the sub-domains takes place in loops over level sequences. One loop sequence of the LTS synchronization is sketched in Fig. 18 for three sub-domains with different time step sizes ($\Delta t_A > \Delta t_B > \Delta t_c$) and levels L_i.

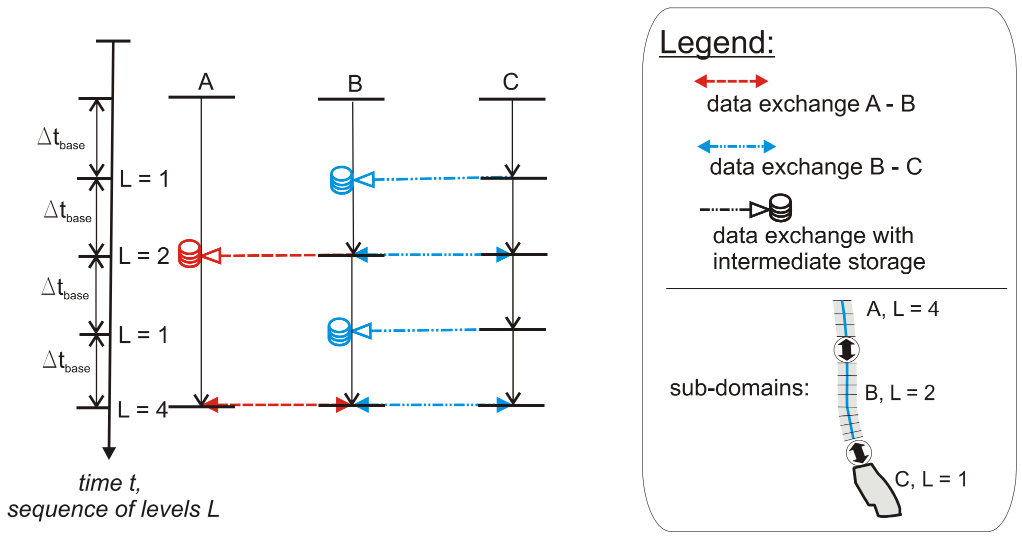


Fig. 18: LTS-synchronization for 3 sub-domains with different time step sizes. The sub-domain C with the smallest time step size determines the base time step. Sub-domains A and B run for multiples of 4 and 2 of the base time step size.

For example, in case that the maximum level of a sub-domain is 8, a level sequence of m = [1,2,1,4,1,2,8] is executed, where m equals the present level of the loop. Each sub-domain is executed only if its level L_i is smaller or equal to the present level m. These sub-domains are then advanced for a time step size of $\Delta t_i = L_i \Delta t_{base}$. Data exchange between adjacent sub-domains takes place only when the sub-domains have reached a common level. If adjacent sub-domains have different time levels then the exchanged data must be stored intermediately to guarantee conservation principles. The data is finally passed over when the sub-domains reach a common time level. After the end of the loop of the level sequence, all sub-domains have been executed at least once and have finally have reached a common final time $t_{new} = t_{old} + n\Delta t_{base}$. From this starting point the levels L_i are assigned again to the sub-domains and the procedure is repeated.

The selection of the base time level is done at the beginning of each level loop. To account for the possibility that the minimum time step could change during the loop iterations, due to changed flow conditions, the base time level can be reduced by a factor $F \leq 1$ for stability reasons.

This page has been intentionally left blank.

6.6 External Coupling

6.6.1 Introduction

The term “external coupling” means the coupling between the program BASEMENT and an external program. This may be e.g. a rainfall-run off model which delivers input data for a river reach or it may be a standalone groundwater model which makes use of the stream water elevations computed by BASEMENT. As described in the model coupling section, one can also distinguish here between one-way coupling and two-way coupling.

One-way coupling
Two different scenarios can be distinguished here:

- An external Program may receive data which is sent by BASEMENT. Therefore the external program must be defined as an external sub-domain in the command file. In the OUTPUT block an output must be defined and connected with this external sub-domain. Wile executing, BASEMENT sends data as soon it is calculated using the output routines to the specified external sub-domain. The external program must fetch the data using TCP/IP routines and must take care of the synchronization, i.e. it must always wait until new data is available.
- Another scenario is to send input data to BASEMENT. Thereby the external program again has to be defined as an external sub-domain in the coupling process and it must be connected with other sub-domains using boundary conditions. Then, the external program can send its data in the XML format to BASEMENT using TCP/IP. BASEMENT takes care of the synchronization within the coupling process and always waits until new data is available.

Two-way coupling
It is also possible to couple an external program with BASEMENT with mutual data exchange. Again, the external program must be defined as an external sub-domain in the coupling process. Furthermore, the external program must implement a synchronization mechanism in order to check if the needed data is available. For the two-way coupling the “local time stepping” can be used as described in the previous chapter. Thereby BASEMENT finally determines the time steps for all sub-domains, including the external sub-domain, in a way that all time steps are multiples of a minimum base time step. Using two-way coupling is sensitive to dead locks and may need some experimenting.

Note
Please be aware that the external coupling approach is still in an experimental stage. Also, the usage of this coupling requires programming efforts and knowledge in TCP/IP programming and XML parsing. External coupling may require the implementation of special boundary conditions in the BASEMENT model. For example, coupling with a groundwater model requires leakage boundaries for water exchange to be set. If you want to make use of such a coupling type, you may contact the developer team regarding the implementation of appropriate boundary conditions in the model.

6.6.2 Data exchange over TCP/IP

The data exchange between BASEMENT and the external program takes place using TCP/IP communication. This has the advantages that it is generally faster than communications via files and enables the coupling between different computers via intra- or internet, even using different operating systems.

Create connection

The communication requires an IP-address and a port-number as identifier and takes place using TCP-sockets. Usually BASEMENT runs as the server application and must be started first. Then, it waits for an incoming connection request. After the incoming request, a connection is established with the external program and the connection information is sent to the external program (socket descriptor). In case of multiple external programs, BASEMENT waits until all connections are established before it starts the computations.

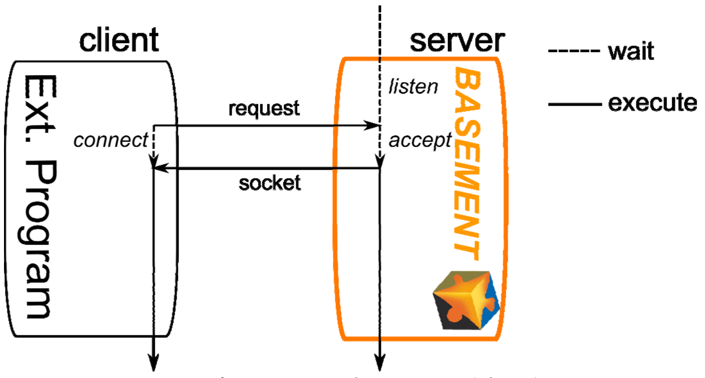


Fig. 19: Connection request from external program (client) to BASEMENT (server)

Data packet

The data is wrapped in “data packets” using the common XML-format, whereby the data values and several additional attributes must be specified. All communications between the programs take place by sending data packets. Therefore also additional information, like e.g. the time, or the time step size, must be included in the data packet. It is also possible to send or receive multiple data packets for different data types or boundaries. The XML-tag has the following structure

<Data *attribute1*=”…” *attribute2*=”...”>…<\Data>

The data values within the XML tag can either be written as ascii or binary data. If ascii format is used, a semicolon separates multiple data values from each other, e.g.

<Data>10.0;10.0;10.0</Data>

The following attributes can be set:

Attribute	**Values**	**Obligatory**
size	number of data values	Yes
time	time of data values in sec	Yes
type	[Q,h,v,…] Type of data	Yes
encoding	[ascii, binary]	No (default = ascii)
byte_order	[little_endian]	No
boundary	name of boundary condition of data values	Only if data is sent to BASEMENT
timestep	current time step of model in sec	Only for local time stepping

Data communication

The data communication via sockets can be compared to data exchange via file-streams. The data packets are inserted into a pipe and the other side of the connection reads the contents after the FIFO concept (First In First Out). The receiving part of the connection must parse the contents of the pipe and extract the data packets. The time attribute of the data packets indicate the time level of the other program required for the synchronization. If the time levels of the data packets are behind the program's time or if no data is in the pipe, than the program must wait and continuously check for incoming data. In case of 'local time stepping' the data packets also carry the information of the local time step size which has to be used.

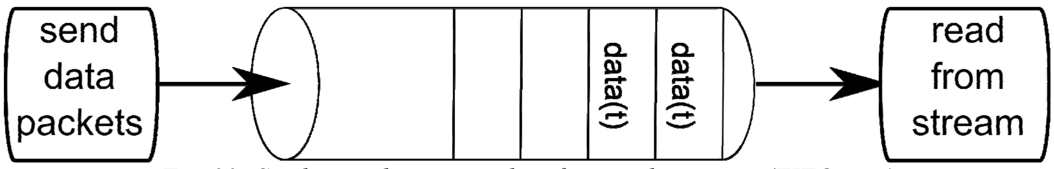


Fig. 20: Sending and receiving data from socket stream (FIFO pipe)

This page has been intentionally left blank.

7 Flow Control in River Systems

7.1 Introduction

The flow characteristics of a river system are not only governed by the character of a channel, the morphology and topography, but also by regulations for hydropower stations and lakes. Such regulations commonly demand that a certain water level is maintained or impose certain limits on the maximum discharge. The exertion of control structures has commonly a significant direct impact on certain river sections or even on the whole river system. The setting of the control structures over time cannot be defined in advance, but depends on the reaction to a change of the whole river system.

The numerical simulation of regulations is very helpful to properly judge such river systems, as it allows assessing and optimizing the effect of individual regulations of control structures on the whole system. This is of great importance as efficient flood control demands an optimal use of existing retention structures.

Therefore, the automatic steering of control structures has been added to BASEMENT, covering 1-D and 2-D simulations as well. The chosen approach allows the simultaneous combination of different controlled and manipulated variables. Controlled variables can be either water surface elevations or discharges. Here not only fixed values can be defined, but also series in time or values depending on the current flow in the river system. As manipulated variable, settings of weir or gates and an abstract outflow hydrograph has been implemented.

Within the present implementation, the determination of the control structure settings have been strongly abstracted, which allow a very flexible integration of further controlling algorithms in the future. As reference, a classical Partial-Integral-Differential (PID) controller has been implemented. By combining various control and manipulated variables within a single controller, BASEMENT now offers the possibility to simulate complex series of weirs over coupled regions.

This page has been intentionally left blank.

7.2 Concept of Flow Control

There are many cases where the behaviour of boundary conditions such as weirs or gates depends on the actual state of the river system and cannot be described by a simple time-dependent boundary setting. An example would be adjusting the weir height in order to maintain a specific water level in front of the weir. This process is commonly denoted as *controlling.* The basic controller has a *controlled variable*, such as water surface elevation, which is desired to be kept at a certain level, i.e. its *target value.* The deviation from the current value of the controlled variable and its target value, also denoted as *error*, is then fed into the controller. The controller reads the deviation and calculates the new value for its *manipulated variable*. An example for a manipulated variable would be a weir height. The new value for the manipulated variable is then fed into the system, i.e. the hydraulic simulation, which finally affects the controlled variable.

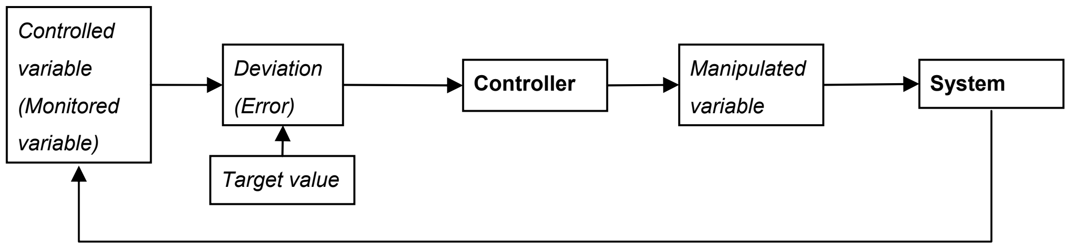


Fig. 21: Basic Control Cycle

In Basement, the *system* is represented by the simulation, the *controller* is a mathematical function $\mathbf{f}(.)$, determining the values of the *manipulated variables* $\mathbf{u}(t)$ from the values of the *monitored variables* $\mathbf{m}(t)$. This can be expressed using the following mathematical expression:

$$\mathbf{u}(t) = \mathbf{f}(\mathbf{m}(t))$$

Logically, there can be multiple monitored and manipulated variables.

7.2.1 Monitored Variable

A *monitored variable* is defined by

$$m_i(t) = v_i(t - \tau_i) - v_{\text{target},i} \,.$$

Here, v_i can be either a water surface elevation, measured on a specific cross-section (1-D) or element (2-D), or a water flow over a cross section (1-D) or a STRINGDEF (2-D). $v_{\text{target},i}$ describes the target value, or in case of a *feed forward* controller, the equilibrium state. τ_i is a delay time controlling when the information of the measured variable is fed into the controller.

7.2.2 Manipulated Variable

A *manipulated variable* refers to a boundary condition and can be a weir height, a gate level or an outflow (in case of 1-D hydrographs as downstream boundary).

This page has been intentionally left blank.

7.3 Controller Types

7.3.1 PID-Controller

One possible approach to describe the mathematical function $\mathbf{f}(.)$ is a PID (partial-integral-differential) controller. This type of controller relates a monitored variable to a manipulated variable by three additive controller elements:

$$u_i(t) = \underbrace{K_{\mathrm{P},ij} m_j(t)}_{\text{P-Element}} + \underbrace{\int_0^t K_{\mathrm{I},ij} m_j(t') dt'}_{\text{I-Element}} + \underbrace{K_{\mathrm{D},ij} \frac{d}{dt} m_j(t)}_{\text{D-Element}}$$

Internally, the PID-controller is implemented in its differential form (i.e. the change of u_i is calculated in each time step). The three required variables are K_P, K_I and K_D. The correct definition of these variables is very crucial to the proper operation of the controller. Which values should be used is highly dependent on the system and therefore requires some care and experience.
The P-element represents an adjustment proportional to the deviation and therefore only limits the deviation, but does not bring the system back into the state where no deviation exists. For this reason, the I-Element integrates the deviation and consequently, the system can be forced into its equilibrium state. If the response of the I-Element is too strong compared to the P-Element, the system oscillates. If the values of both P and I elements are too small, the reaction of the system is very slow or even too weak to re-establish the given targets. The D-Element depends on the change of the monitored variable and is used to quickly adapt the manipulated variables in case of a fast change.

More on the determination of correct PID coefficients can be found in the article by Fäh and Kühne (1987). Recommendations on how to choose the coefficients are given in the integrated help of the software.

This page has been intentionally left blank.

References

Beffa, C., (2002). „Integration ein- und zweidimensionaler Modelle zur hydrodynamischen Simulation von Gewässersystemen“*. Int. Symposium Moderne Methoden und Konzepte im Wasserbau*, ETH Zürich (Oktober 2002).

Chandra, R., Dagum, L., Kohr, D., Maydan, D., McDonald, J., Menon, R., “Parallel Programming in OpenMP”

Chapman, B., Jost, G., van der Pas, A., (2008) “Using OpenMP - Portable Shared Memory Parallel Programming”, *MIT Press*

Fäh, R., Kühne, A., (1987). “Wasser, Energie, Luft – eau, énergie, air”, Heft 5/6, p. 93 (1987).

Miglio, E., Peretto, S., Saleri, F., (2005) “Model coupling techniques for free surface flow problems: Part I”, *Nonlinear Analysis: Theory, Methods & Applications,* 63 (2005) 1885-1896.

Kuhn, B., Petersen, P., Kuck & Associates Inc. OpenMP versus Threading in C/C++

Manferdelli, J.L., Govindaraju, N.K., Crall, C., Challenges and Opportunities in Many-Core Computing. *Proceedings of the IEEE* (May 2008)

Online reference of OpenMP, freely available at http://www.openmp.org

Sanders, B., (2008). “Integration of a shallow water model with a local time step”. *Journal of Hydraulic Research*, 46(4), 466-475.

This page has been intentionally left blank.

PRE PROCESSING

of BASEMENT

This page has been intentionally left blank.

Table of Contents

1 General

1.1 Requirement 1.1-1
1.1.1 Project / Scenarios 1.1-1
1.1.2 Input data 1.1-1
1.1.3 Boundary conditions 1.1-2
1.1.4 Source terms and local properties 1.1-2

2 Model input data

2.1 Topographic data sources 2.1-1
2.1.1 Terrestrial data or surveying with differential GPS 2.1-1
2.1.2 Official topographic survey 2.1-1
2.1.3 Laser scanning 2.1-2
2.1.4 Sources and quality of data 2.1-3
2.2 River related data sources 2.2-1
2.3 Characteristic quantities of riverbed 2.3-1
2.4 Processing the raw data 2.4-1
2.4.1 Topography 2.4-1
2.4.2 Hydrology 2.4-1
2.4.3 Grain size distribution 2.4-2
2.5 GIS Interface 2.5-1
2.5.1 Administration of geographical data 2.5-1
2.5.2 Plane assignment for point data 2.5-1
2.5.3 Interoperability and standard interfaces 2.5-1

3 Grid generation

3.1 Introduction 3.1-1
3.2 Topographical input data 3.2-1
3.3 Mesh quality 3.3-1
3.3.1 Shape of mesh elements 3.3-1
3.3.2 Ambiguous gradients 3.3-2
3.4 Issues on triangulation 3.4-1
3.4.1 Delaunay triangulation and constrained Delaunay triangulation 3.4-1
3.4.2 Algorithms 3.4-3
3.5 Use of the grid generator of SMS 10 .... 3.5-1

List of Figures

Fig. 1: *Visualization of the vertical setup for a computational cell within the models................2.3-1*

Fig. 2: *Unstructured grid: Triangulation 3.1-1*

Fig. 3: *Left: ambiguous quadrilateral element with false break line; right: correct discretisation of the dyke crest. 3.3-2*

Fig. 4: *a) Empty circle criterion satisfied. b) Empty circle criterion not satisfied 3.4-1*

Fig. 5: *a) edge circle criterion. b) empty circle criterion...3.4-2*

Fig. 6: *a) triangulated half planes. b) merged triangulation..3.4-3*

Fig. 7: *a) point insertion and determination of affected triangles. b) new triangulation 3.4-4*

Fig. 8: *a) insertion of the new point b) edge swapping based on empty circle criterion.............3.4-4*

Fig. 9: *Simulation procedure with use of SMS 3.5-3*

List of Tables

Tab. 1: *Accuracy of current measurement methods 2.1-3*

Tab. 2: *Products offered by “swisstopo” 2.1-4*

Tab. 3: *Products offered by Swissphoto AG...2.1-4*

II PRE-PROCESSING

1 General

1.1 Requirement

The main purpose of the pre-processing activities is to define the project and the different scenarios, to prepare the available (topographic) input data and to put it in the format needed by the computational module as shown in the overview in part I of the User Manual. Additionally, the choice of computational approaches and boundary/initial conditions has to be made.

1.1.1 Project / Scenarios

A project is defined for one region which is to be analyzed. Within a project, one or more scenarios can be embedded. To manage a project, it is therefore necessary to define which scenarios, files, and other elements belong to it. This includes the information where all these elements are stored and how they are connected to each other.
For the same project, several scenarios can be created. For each of them a simulation is executed. The different scenarios can vary regarding the computational mesh, other input data, the used approaches and the boundary conditions. In addition, type, location and time of the results to save have to be specified. The simulation can be done in 1, 2 or 3 dimensional computation or as a combination of them. The phenomena to be considered have to be chosen as well, as there are at the moment: dam break, bed load, suspended load, debris flow, mobile bed, erosion, collapse, pollutants etc.

1.1.2 Input data

As mentioned in the introduction, there are three main types of data to be provided for a simulation: topography, hydrology and sediment data. All data has to be transformed in a certain way to satisfy the input specifications of the main computation program. The precise specifications are available in the reference manual.

1.1.2.1 Topography / Computational mesh

The most important, and most laborious to achieve, input is the retrieval and setup of the computational mesh. It is based on the real world topographic data. At the end of the pre-processing task, a grid in a defined shape and format is available for the simulation module.
This mesh can be generated in different ways. The topographical raw data may come from a cluster of points described by three dimensional coordinates, digitized contour lines, break line polygons or cross sections and probably is furnished in different file formats, which have to be interpreted and transformed. For a 2d simulation, in a first step a TIN (Triangulated Irregular Network) has to be generated. Then this mesh has to be modified and refined in order to satisfy the special mesh qualities needed for stability of computation.

For the 1d model it might be necessary to interpolate cross sections or deduce cross sections from a DEM (digital elevation model).

1.1.2.2 Hydrologic data

At all boundaries of the mesh, hydrologic boundary conditions such as hydrographs or time series of water levels have to be provided. In the case of a simulation with sediment transport, also a sediment concentration in the water might be needed.
The choice of boundary data has to be made with care, considering the type of event being simulated. The data might be created especially for the wanted simulation hypothesis or be adapted from existing measured or statistical data. Some times, special manipulation of the data turns out to be necessary, for instance to omit discharges not relevant for sediment transport. Finally, again, the hydrologic data has to be put in the format wanted by the program.

1.1.2.3 Sediment data

Sediment data primarily comes from water sediment samples or surface samples like the line method. Possibly, a grading curve still has to be built from this raw data. Then the number of desired grain classes for the simulation has to be chosen and the characteristic grain classes need to be identified. Based on the grain distribution, other values like roughness or angle of rest may have to be calculated.

1.1.3 Boundary conditions

The boundary conditions define quantities at the margin of the computational region during the whole simulation time. These are typically hydrographs at the inflow boundary and water levels at the outlet boundary. In general, different special conditions can be applied by defining for instance the water level of a lake, weirs, steps, etc.

1.1.4 Source terms and local properties

For every mesh element, several characteristics can be specified, for example not flown through cells, mobile bed, bed load potential, k-value, roughness, shear stress or others. Sources and sinks of water and/or sediments might be added within the computational region.

2 Model input data

2.1 Topographic data sources

The topographic raw data in the form of digital terrain information builds the fundamentals for grid generation and further numerical simulations using the BASEMENT software. In the case of 1d simulation, the raw data consists of recordings of river cross-sections. In a 2d or 3D model, the raw data is built from point clouds, height contour lines or a digital terrain model (DTM).
Dependent on the assignment and its requirements, most of the raw data usually gets collected by experts. However, there are some extensive topographic models with different quality available, see e.g. the cross-sections database from the Federal Office for the Environment (BAFU) and high resolution terrain models from “swisstopo” or “Swissphoto AG".
The next sections provide a short overview of different methods and data sources of data sources for topography and also hydrology and sediment. Most data sources in the next few sections are somehow related to Switzerland. Other regions and countries provide similar data – depending on the actual law.

2.1.1 Terrestrial data or surveying with differential GPS

Surveying or special engineering agencies provide terrestrial terrain data or terrain data obtained by differential GPS (DGPS) of the desired quality. A pure terrestrial recording is more expensive because at least two workers are needed in contrast to a DPGS scan, which can be done by one employee only.
The survey with GPS needs a certain visibility concerning the GPS satellites. The accuracy depends on the exactness of positioning for the reference point but usually lies in the range of a few centimetres.
The terrain data is delivered in different formats as e.g. xyz-coordinates or GEOBAU standard. Be sure to specify your claims on quality of the data as resolution, typical terrain deformations but also the desired data format when asking for a tender offer.

2.1.2 Official topographic survey

According to a decision of the Swiss federal assembly (“Einführung der amtlichen Vermessung” AV93), the bureau of land charge register provides digital cadastral data almost over the whole country (ca 80%) and data from the country’s information system (ca 30%).
The cadastral information is just of secondary use as the parcels are all flat. However, the ground plans of certain constructions and buildings may be helpful.
The data from the country’s information system can be used (with some additional work) for the definition of boundary conditions, e.g. data about floor cover types.
The data format used for the AV93 act is called INTERLIS (Data exchange mechanism for “Landes-InformationsSysteme”).

2.1.3 Laser scanning

Essential factors for an accurate triangulation concerning topographic raw material are accuracy of position and height as well as the density of the measured points. Of special interest are therefore height model data taken by Airborne Laser Scanning (LIDAR) as e.g. provided by "swisstopo". This method allows for a good data base within forests. Only in dense coniferous forest, the determination of the terrain model tends to be impossible. Of course, as with all measurements from the air, neither ground nor surface levels from water bodies can be obtained.
As the surface model generated from the LIDAR method distinguishes between terrain, buildings and vegetation, it provides a good starting point for further processing towards a numerical grid.

The federal office for topography ("swisstopo") has commissioned Swissphoto AG to collect height data all over Switzerland for the identification of agricultural areas. However, this project is restricted to areas below 2000 m asl. For higher located zones (which could be relevant for debris flows), the data could be produced with reasonable costs. Until end of 2007 this project shall be finished.

"swisstopo" delivers their terrain data in two formats:

- DTM-AV raw: a point cloud with averagely 1 point per 2 m^2
- DTM-AV grid2: an interpolated 2x2m mesh

The accuracy of position and height is about +/- 0.5 m. Additionally, "swisstopo" also works with 1x1m meshes which reach an accuracy below +/- 0.3 m.

For large areas and a high density of grid points, the amount of data is beyond the resources even of nowadays computing power. Therefore, there exists a variety of algorithms which eliminate all unnecessary points in a certain area that have no influence on the shape of the triangulation. This may reduce the amount of data by 10 – 60 %.

2.1.4 Sources and quality of data

One main factor for the quality of the results is the accuracy of the original topographic data. The accuracy varies depending on the objective of the recording, as well as on the recording methods. The accuracies of the most diffused methods are listed in *Tab. 1.*

	Situation accuracy	**Height accuracy**	**Observations**
Terrestrial	+/- 2 cm [1]	+/- 5 cm [1]	
Sonic depth finder	+/- 5-10 cm [1]	> 10 cm [1]	
GPS	+/- 1-3 cm [1] +/- 1 cm [4]	+/- 5cm [1]	At least 4 satellites
Orthophoto		+/-20-3 cm [1] 0.2 – 0.3 ‰ of flying altitude[3]	3 cm with photo scale 1:2000
LiDAR		+/-20-30 cm [2] open lea +/- 15-20 cm forest insecure [3]	
Radar (InSAR)		open lea +/- 20 cm[3]	

Tab. 1: Accuracy of current measurement methods

[1] Gewässergeometrie, Landesanstalt für Umweltschutz Baden-Württemberg, Karlsruhe 1999.
[2] Swissphoto AG (verbal)
[3] Leitfaden Qualitätssicherung Photogrammetrie und DTM-Generierung, Konferenz der Kantonalen Vermessungsämter, 2000
[4] GPS global positioning System, "swisstopo"

The products which are directly available in Switzerland and their accuracies are listed in the following tables.

Product	**Density**	**Source**	**Format**	**Accuracy**
DHM25 Base Model DTM	35-1600 point/km^2	1:25000 map vectorized contour lines and contour lines in lakes, lake perimeters and spot heights, main alpine break lines (photogrammetric data)	Arc View Shape-Files, DXF,GEN, BMBLT	Mean error 1.5 to 3 meters depending on the regions
DHM25 Matrix Model DTM	1600 points/km^2.	interpolation of the Base Model on a 25x25m grid	MMBLT, MMBL, AIGRID, XYZ, DXF, VRML	
DTM-AV raw	1 point for 2 m^2	based on official measurement with Lidar Airborne Laser Scanning	ASCII , Interlis, others possible if requested	height accuracy +/- 50 cm
DTM-AV grid2		2x2m grid interpolated from DTM-AV raw		
DSM-AV raw	1 point for 2 m^2	with buildings and vegetation	ASCII , Interlis, others possible if requested	height accuracy +/- 50 cm, +/-150cm for vegetation
DSM-AV grid2		2x2m grid interpolated from DSM-AV raw		

Tab. 2: Products offered by "swisstopo"

Product	**Density**	**Source**	**Format**	**Accuracy**
DSM	10x10m, 20x20m, 50x50m	Photogrammetric interpretation of aerial photos from 1995/96	ASCII xyz or GRID	height accuracy: 3-5 m midland, 7-10 mountains
DEM LIDAR (on commission)	31000/(250x 160 m), spacing 1-1.3m	Lidar Airborne Laser Scanning	ASCII xyz ASCII ArcInfo, …	Height accuracy 30 cm

Tab. 3: Products offered by Swissphoto AG

2.2 River related data sources

The following list provides some more Swiss data sources for real world problem modelling and boundary conditions.

River cross-sections:

Primary use for 1d models; also qualified for 2d models to improve topography data within a rivers area. Terrestrial mapping, optimal quality and format according to the cross-section database of the Federal Office for Environment FOEN (formerly FOWG / bafu.admin.ch)

Terrain topography:

Basic dataset for simulations with spatial (2- or 3d) models. Optimal quality and format (ASCII x y z) as swisstopo DTM-AV / DOM-AV (New swisswide high resolution DTM/DOM based on laser scanning)
(http://www.swisstopo.ch/de/digital/dom.htm)

Surface texture and special buildings:

Declaration of roughness coefficients, porosity and erosion resistance of surfaces; Optimal format INTERLIS-1, as e.g. DOM-AV or floor cover plans AV93 (generated by the cadastral register on demand of the cantons)

Sedimentological /geological data:

Surface characterization of the riverbed (mostly using line samples), of the erosion capable underground (volume sampling or geological drilling) and of the flow induced transported material (grain size distribution from sieve analysis)
Possible data sources: geology of cantons, FOEN (formerly BWG), Nagra, in situ suspended load and bed load measurements.
The estimation of sedimentologic data and its analysis is surely the most time consuming part. At certain water bodies, a continuous monitoring of suspended load exist. However, the granulometric size distribution is seldom measured. Therefore, the bed load near the surface but also the suspended load often have to be estimated under certain assumptions or they will have to be measured.

Hydrological data:

Temporal or stationary boundary conditions for the numerical model, e.g. inflow discharges, water levels and local sources or sinks.
Possible data sources: Hydrological data from the federal measuring facilities, specific rainfall-discharge models (e.g. IFU/ETHZ), retention models.
For larger water bodies within Switzerland (observed by the FOEN), time series and hydrographs are available online. Smaller watercourses are often not covered and need either a hydrologic model or new measuring/monitoring to determine the discharge.

This page has been intentionally left blank.

2.3 Characteristic quantities of riverbed

The numerical models used for the computation of hydraulic behaviour are always declared to be either 1d (flow in direction of main flux / x-axis) and/or 2d (horizontal, depth averaged flow field). However, the fundamental terrain topography is always three-dimensional. The spatial discretization of the transport equations is based in 1d on cross-sections or in 2d on an unstructured grid of mostly triangles. The vertical component consists of different layers separated by a planar joint face for both, 1d and 2d simulations.

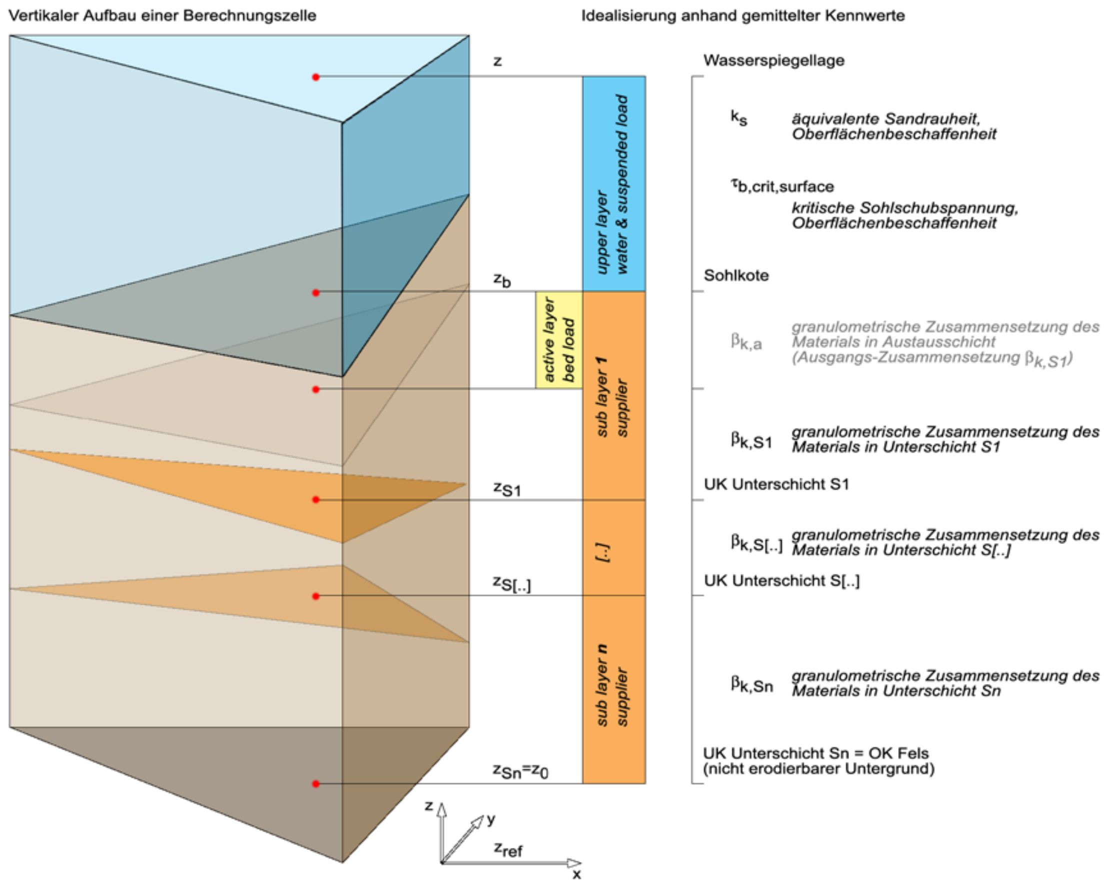

Vertikaler Aufbau einer Berechnungszelle
Idealisierung anhand gemittelter Kennwerte
z
Wasserspiegellage
k_s äquivalente Sandrauheit, Oberflächenbeschaffenheit
$\tau_{b,crit,surface}$ kritische Sohlschubspannung, Oberflächenbeschaffenheit
upper layer water & suspended load
z_b
Sohlkote
active layer bed load
$\beta_{k,a}$ granulometrische Zusammensetzung des Materials in Austauschschicht (Ausgangs-Zusammensetzung $\beta_{k,S1}$)
sub layer 1 supplier
$\beta_{k,S1}$ granulometrische Zusammensetzung des Materials in Unterschicht S1
z_{S1}
UK Unterschicht S1
[..]
$\beta_{k,S[..]}$ granulometrische Zusammensetzung des Materials in Unterschicht S[..]
$z_{S[..]}$
UK Unterschicht S[..]
sub layer n supplier
$\beta_{k,Sn}$ granulometrische Zusammensetzung des Materials in Unterschicht Sn
$z_{Sn}=z_0$
UK Unterschicht Sn = OK Fels (nicht erodierbarer Untergrund)
z
y
x
z_{ref}

Fig. 1: Visualization of the vertical setup for a computational cell within the models.

The characteristic values are indicated relative to a joint surface (e.g. riverbed level) or to a layer (e.g. granulometric composition of bed material in layer S1). Their properties are represented in the balance point of the corresponding joint plane. This results in a simplified model of the layering for a computational cell. The characteristic data used in the model to describe the state and conditions of the surface are defined as followed:

Position:

Soil – properties

z_B	bottom elevation (bed level)
z_{Sub_i}	elevation level of the lower, closing joint plane of the corresponding sublayer Sub_i $(i \in \{1 \ldots n\}$, with $n =$ maximum number of Sublayers)
z_0	elevation level of the lower joint face of sublayer Sn, which finishes the model downwards, respectively describes the position of the non erodable underground.

(all variables are measured in [m.a.s.l.] or in [m] relative to a userdefined reference system)

Attributes:

Vegetation – ground covering

k_s	[m]	equivalent sand roughness based on Nikuradse: Description of the surface roughness (e.g. grass, sealed plane, etc.). (default value or derived by grain distribution of the first sublayer)
$\tau_{B,crit}$	[N/m^2]	critical value for begin of sediment movement at the surface, in the sense of a rised erosion constancy, e.g. by natural cover (default values)

Soil – properties

β_{g,Sub_i}	[-]	mass fraction of the g^{th} grain class relative to the total mass of sublayer Sub_i (default values according to the grain size distribution)
β_g	[-]	mass fraction of the g^{th} grain class relative to the total mass of the active layer (possible default values according to line samples if available)

The listed attributes are mostly not completely determinable directly. For example, the equivalent sand roughness k_s based on Nikuradse is an experimentally obtained value for the flow resistance of different kind of surfaces. In the case of a more complex soil structure, k_s may consist of several components. Its actual value has to be determined by a calibration of the numerical model. The same holds true for the critical value $\tau_{B,crit}$ which defines the beginning of movement at the soil surface. This value is affected by the character of the bed surface like natural cover or sealing (at farmlands). Without special conditions, the critical value can be estimated via the grain size distribution.

The mass fraction β_{g,Sub_i} of grain class g in sublayer Sub_i has to be derived from an experimental grain size distribution using a sieve analysis. The material to be sieved may come from a near surface sample take (e.g. by daggering) or a drill hole. The granulometric composition directly at the surface can be estimated by a line sample. Accordingly, the corresponding mass fractions β_g in the active layer can be determined.

2.4 Processing the raw data

It was shown in the previous sections that the numerical model needs some specific characteristic values which usually are not given directly by in situ measurements. Often, some experience in handling computational simulation models is necessary to reproduce the natural conditions in an adequate way and obtain realistic results.

2.4.1 Topography

The topographic raw data for 1d or 2d simulations comes in the form of cross-sections or by a digital terrain model (DTM). For a 1d model, cross sections are mostly of accurate quality and need just few corrections and adaptations. However, most DTM's serving as a base for a 2d model need an elaborate revision because of the surface triangulation:

- the resolution of the DTM and the computational grid are not congruent (e.g. areas of lower interest shall have a coarse grid to reduce computational effort – the DTM usually has a uniform point density)
- the course of water bodies is not continuous or cross sections are not complete (e.g. data only available up to the water level)
- passages are not omitted (e.g. bridges)
- modelling and representation of buildings is not a priori known (e.g. height of buildings, retention volume, flow resistance)
- relevance of waste edges and artefacts are costly to determine (e.g. small walls, temporary dumpsites, hedgerows)
- etc.

The choice of software resources for processing the topographic raw data is huge, but often, these programs are somehow limited to specific applications and data formats. It is therefore recommended to specify the desired data format and resolution in advance to avoid unnecessary repetitions.
More detailed information about grid generation is given in the next chapter.

2.4.2 Hydrology

The hydrologic raw data for discharge and water levels usually has a non uniform resolution in time. Often, the measurements are indexed with date and time. For the simulation module, the time series have to be converted to a system with standard units (e.g. seconds). In an easy case, the converted data can directly be used as a boundary condition. The simulation module will then interpolate the desired values to the actual computational time.
Further modifications may be necessary if e.g. just some time slots have to be simulated or if the main processes for a phenomenon like sediment transport occur at no more than a certain water discharge.

2.4.3 Grain size distribution

As mentioned earlier, the granulometric composition of the erodable material gets measured by a line sample or a sieve analysis and results in a grain size distribution (grain diameter versus weight percentages relative to the total sample weight). For the simulation, this distribution has to be discretized and desired grain classes have to be determined. Each grain class is defined by its medium diameter and has a percentage. The choice of the numbers of grain classes and thus the resolution of the material composition depends on the size of the problem and the computing power available. The classification of the grain diameters which represent a grain class strongly affects the transport behaviour of the whole material and the efficiency of the transport model.
It is recommended to run several simulations with one but also with more grain classes and to compare them.

2.5 GIS Interface

A geographic information system (GIS) as assistance for a numerical simulation model shall provide several data in best possible spatial and temporal resolution but not necessarily in a model inherent format. The data structures described in the following are just proposals. For all of them, one has to decide a priori whether the raw data shall be saved per se or in a processed form (raw data and pre-GIS-processed data).

2.5.1 Administration of geographical data

Generally, one distinguishes two kind of data types: matrix data (based on a uniform matrix, e.g. air taken pictures) and vector data (e.g. property borderlines).
Vector data serves for the visualization of the geometry and uses different data structures. Objects with simple geometry (see "OpenGIS Simple Feature Specifications") are e.g. points, lines and polygons. For complex geometries (see "OpenGIS Geography Markup Language (GML)"), additional information as topology and other attributes (e.g. time) are required.
A further possibility to structure geographical data in a GIS is a splitting on different topic layers. Digital terrain models with height information are represented in a GIS either as TIN (triangulated irregular network) or as a GRID (structured mesh). With regard to numerical simulation models, the topology of a TIN or a GRID corresponds to unstructured resp. uniform computational grids.

2.5.2 Plane assignment for point data

In a GIS, real, area-related conditions are given in an idealized way. On the one hand, there is a limited number of attributes and on the other hand the values over the surface are averaged. Partially, the data base is just point wise available and just sparsely against resolution and process scales of numerical simulation models, e.g. at geological probe holes. Therefore, the coarse values have to be extrapolated on the corresponding surfaces of the numerical grid. The determination of the corresponding surface, or i.e. the surrounding polygon, needs different geometric constructs as e.g. Dirichlet Tesselation (Voronoi-Polygons). Additionally, one should consider related information from superior scales (e.g. geological maps with layering data and downcasts).

2.5.3 Interoperability and standard interfaces

To assure interoperability – meaning system independent communication – between different information systems, several norms for a standardized data representation exist. Examples are the "OpenGIS Geography Markup Language (GML)" or the norms used in Switzerland: INTERLIS (SN 612 030 / 612 031) and GEOBAU (SN 612 020).

This page has been intentionally left blank.

3 Grid generation

3.1 Introduction

The numerical methods used for the approximation of differential equations needed in flow simulations are based on a discretization of the domain in simple small shapes. The complex of these elements forms a mesh. In BASEMENT, the finite volume method, being particularly valuable for fluid dynamics, is used.
To describe a topographic surface, the terrain data is often triangulated to allow a perspective representation of the topography. The result is a piece-wise linear interpolation of the surface. This triangulation can be used directly as mesh for simulations, as it permits to have the original data in the vertices of the initial grid and no interpolation is necessary. However, most of the time, the mesh has to be transformed somehow. Regions of high interest need some mesh refinement for higher accuracy and areas of lower interest are often coarsened to save computing power.

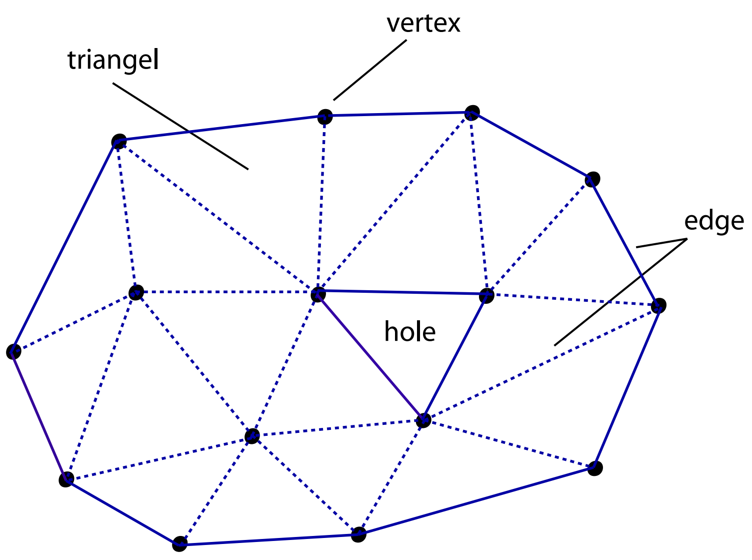


Fig. 2: Unstructured grid: Triangulation

In numerical simulations, two types of meshes are used: structured and unstructured grids.
Structured grids are simpler to use, but need an interpolation of the data to determine the values at the desired vertex position if the input is an irregular point cloud. In addition, they are not sufficiently flexible to fit complex geometries.
Among unstructured meshes, the triangulated irregular networks (TIN) are most convenient because they fit the irregular distribution and complex geometry of the topography best. Furthermore, they allow for a rapid transition from small to large elements and the insertion and conservation of break lines. BASEMENT is built on unstructured grids.
The computational mesh consists of vertices, edges which connect the vertices and elements (control volumes) which are bounded by the edges.

The triangulation of terrain data is a special case of mesh generation, as the vertices do not only have a position in the coordinate system but also elevation information. This is a so-called 2,5 dimensional problem.

3.2 Topographical input data

The original terrain data typically has a very irregular distribution and a locally variable density. Data can be available in different shapes:

- A cloud of points (x, y, z) e.g. digital elevation model;
- Polygons e.g. digitalized contour lines;
- PSLG (planar straight line graph) e.g. a DEM with addition of break lines. This is the most general case, in which the input is a set of vertices and non-crossing line segments that must be conserved in the triangulation.

Although the raw input data, e.g. from a point cloud, could be directly used as a grid, usually further transformations are performed to gain a suitable computational mesh. A suitable mesh is mainly defined by its quality.

This page has been intentionally left blank.

3.3 Mesh quality

3.3.1 Shape of mesh elements

The shape of the elements of the meshes has an important effect on the applicability of numerical methods. Speed (due to convergence time), accuracy and stability of a simulation depend strongly on the quality of the employed mesh. Therefore, it is important to produce the best possible triangulation for the application.

Size, shape and number of the elements play an important role for the quality of the mesh. Possible characterizations of a triangulation, which can be optimized depending on the application, are:

- minimal angle
- maximal angle
- maximum edge length
- total edge length
- maximum height
- area of the triangle
- aspect ratio: ratio of the length of the longest side to the height (definition after (Bern and Eppstein 1995), other similar definitions existent)
- aspect ratio: ratio of the circumference and the in-circle of a triangle
- roughness of the piecewise linear interpolation of a 2,5 D problem (The roughness of an interpolated surface can be measured e.g. by the Sobolev semi-norm)

The criteria which determine the quality of a mesh differ depending on the application (respectively partial differential equation) and the numerical method implemented. Also, the possible optimization depends on the constraints given by the original data. It is recommended to avoid L-shaped overall areas (non convex hull) as such problems often lead to numerical instabilities in the concave corner.

In general, a height aspect ratio, very small angles and especially very large angles are considered as bad, as they lead to a poor numerical condition of matrices and increase the approximation error, which arises with the element size in general, as well. Big differences of size between neighbouring cell elements can have a negative influence on the numeric simulation too.

3.3.2 Ambiguous gradients

A further mesh quality criterion specific to meshes consisting of quadrilateral elements, or meshes with mixed element types, is the problem of ambiguous gradients. Quadrilateral elements are defined by four nodal points. These nodes ideally have elevations which guarantee that the four nodes lie within a plane. But in case of strongly varying terrain topographies, a bad placement of quadrilateral elements can lead to situations where this is not the case. Such elements are deformed and have an ambiguous gradient. *Fig. 3* illustrates a quadrilateral element with ambiguous gradient which is situated across a river dyke.

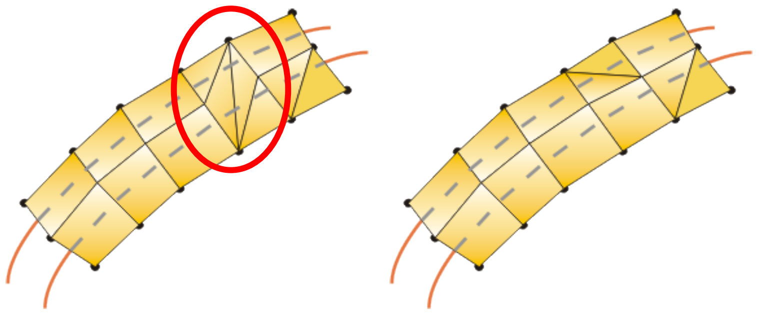


Fig. 3: Left: ambiguous quadrilateral element with false break line; right: correct discretisation of the dyke crest.

In such situations a splitting of the quadrilateral element into two triangles becomes necessary. The selection of the correct break line within the quadrilateral element (here, *along* the dyke crest) must usually be done manually according to local topography. To facilitate this task, most grid generating tools (e.g. SMS) offer special features for detecting quadrilateral elements with ambiguous gradients in the mesh.

3.4 Issues on triangulation

Mesh triangulation and grid refinement play an important role in almost every numerical simulation. Therefore, a lot of different techniques have been developed to achieve suitable computational meshes. The user is basically free to use any tool or method which generates a mesh of accurate quality out of the raw data. This section shall give a short overview on some popular triangulation methods.
BASEMENT does not provide an automated routine for mesh generation. However, at the moment the program supports any input generated by SMS 9.2 (Surface Water Modelling System), a commercial tool specialized on topographic grids for conservation laws which will be discussed shortly in chapter 3.5.

3.4.1 Delaunay triangulation and constrained Delaunay triangulation

The Delaunay triangulation is one of the most employed triangulation methods because it optimizes several quality criteria: It maximizes the minimal angle and minimizes the maximum containment circle radius, the maximum enclosing circle radius and the roughness of a piecewise-linear interpolation. It also provides good results regarding the minimization of the maximum angle, but it does not find a global optimum in this case.

A Delaunay triangulation has the following properties. It

- is the straight line dual of the Voronoi diagram;
- is unique;
- respects the circumcircle criterion;

The circumcircle criterion is respected if the circumcircle of every interior triangle does not contain other points.

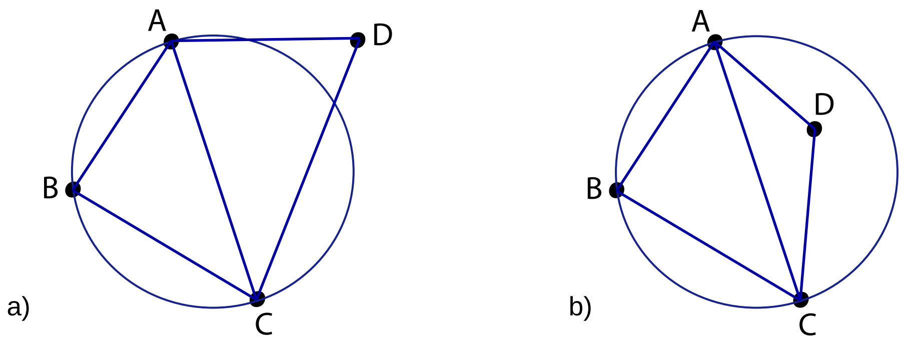


Fig. 4: a) Empty circle criterion satisfied. b) Empty circle criterion not satisfied

This corresponds to say that if ABC + CDA < 180° the empty circle criterion is satisfied.

- respects the edge circle property: for each edge exists some point-free circle which passes through the end points;
- respects the neighbour property: an edge formed by joining a vertex to its nearest neighbour is an edge of the triangulation.

3.4.1.1 Constrained Delaunay triangulation (CDT)

The terrain data is sometimes provided in the shape of a PSLG as it contains break lines which must be conserved as edges in the triangulation. In this case, the constrained Delaunay triangulation can be used.

For the definition of a constrained Delaunay triangulation the notion of visibility of a point is needed: In a PSLG domain P a point D is visible to a point C if the open line segment CD lies within the domain and does not intersect any edges or vertices of P. Point D is visible to CB if it is visible to some point on CB.

For the CDT the edge circle and the empty circle criterion are modified as follows:

- Edge circle: for each edge a circle passing through its end-points containing no other point of the domain visible to the edge exists;
- Empty circle: the circumcircle of every triangle contains no points visible to points inside of the triangle.

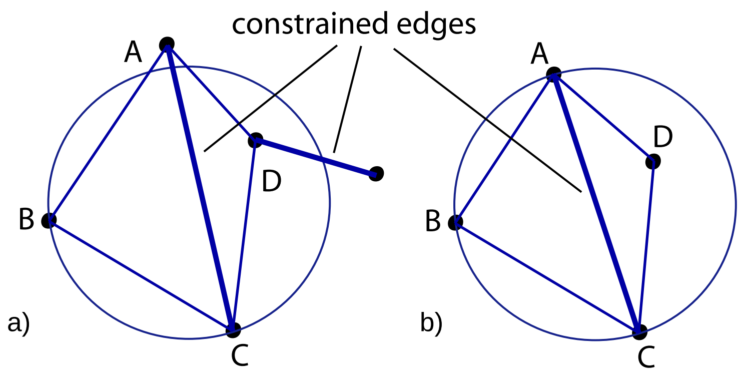


Fig. 5: a) edge circle criterion. b) empty circle criterion

3.4.1.2 MinMax triangulation

The MinMax triangulation minimizes the maximum angle. It has proven to be very useful in CFD (Bath 1995).

3.4.1.3 Data dependent triangulation

A data dependent triangulation can be used for a 2.5 d problem. Its aim is to find the best triangulation under data dependent constraints, by minimizing a local cost function of a piece-wise interpolation. Two examples of local cost functions are described in (Bath 1995).

3.4.1.4 Steiner triangulation

A Steiner triangulation is a triangulation in which extra points are added to the original data to improve the quality of the mesh. The additional points are called Steiner points. The number of Steiner points must be limited, limiting also the quality of the mesh.

3.4.1.5 Non obtuse triangulation

One of the most interesting types of Steiner triangulations is the triangulation without obtuse angles. It is a Delaunay triangulation or constrained Delaunay triangulation and also minimizes the maximum angle and maximizes the minimum height (Bern and Eppstein 1995).

3.4.2 Algorithms

3.4.2.1 Sweep-line algorithm

The sweep-line algorithm can be used to perform a first triangulation without any quality criteria. It adds the points by x-coordinate order and then connects them to all visible points of the convex hull of the existing triangulation.

3.4.2.2 Divide and conquer

For the divide and conquer algorithm the point set is divided in two half planes along the x axis. Then each half plane is triangulated recursively, and finally the two planes are merged.

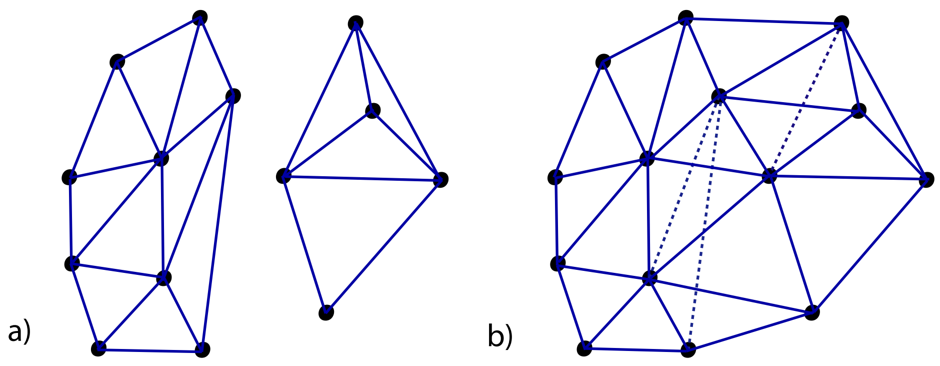


Fig. 6: a) triangulated half planes. b) merged triangulation

3.4.2.3 Incremental Insertion algorithms

The incremental insertion algorithms successively insert new points to an existing Delaunay triangulation. These algorithms have a worst case running time of (n^2) if the points are badly ordered. But in practice it is near to 0(nlogn) for Green-Sibson.

Watson (or incremental delete and build)

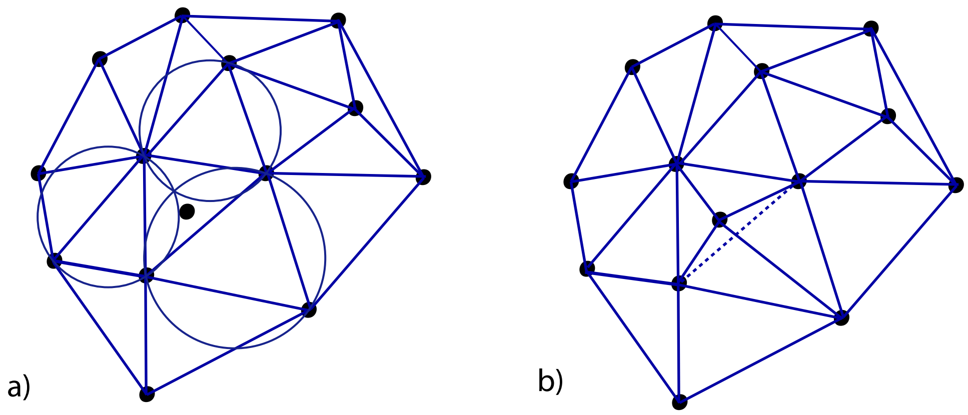


Fig. 7: a) point insertion and determination of affected triangles. b) new triangulation

In this algorithm, after the insertion of the point, all the triangles containing the edge q are searched. Then the edges of these triangles visible to q are deleted and the new edge is connected with the vertices of the originated polygon creating new edges.

Green-Sibson

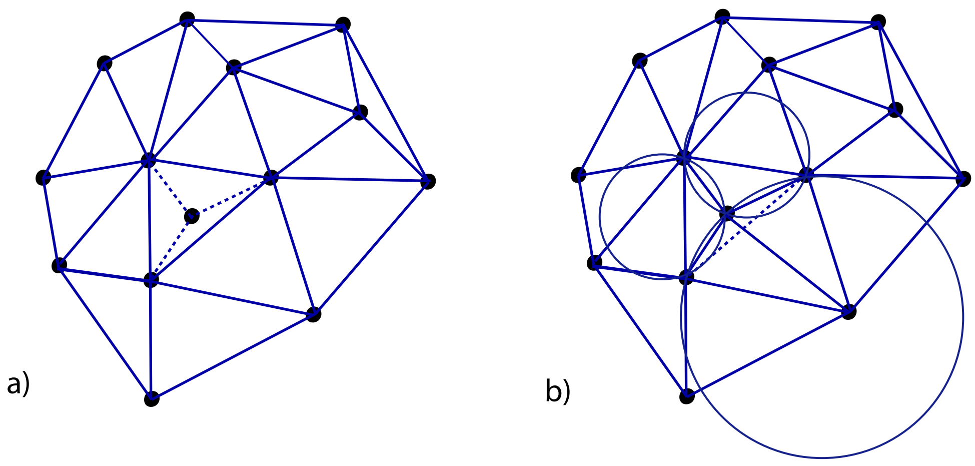


Fig. 8: a) insertion of the new point b) edge swapping based on empty circle criterion

After its insertion, the point q is connected to the vertices of the triangle that contains q.

Then all suspect edges have to be swapped to satisfy the empty circle criterion if we want to obtain a Delaunay triangulation. Other edge swapping criteria can be used, for instance the minimization of the maximum angle.

3.4.2.4 Edge flipping algorithm

The edge flipping algorithm is based on the local optimization of an initial triangulation. For each quadrilateral formed by a convex pair of triangles the diagonal is chosen with regard to a local optimization criterion.

Possible optimization criteria:

- Empty circle criterion: a Delaunay triangulation or constrained Delaunay triangulation is obtained. In this case a global optimum is reached;
- Minimize the maximum internal angle for both triangles: gives a MinMax triangulation only locally optimized;
- Optimize a local cost function: leads to data dependent triangulations (only locally optimal triangulation) (Bath 1994).

Other properties that can be optimized with edge flipping are for instance vertex degree or total edge length, but a global optimum is not guaranteed.
If the algorithm is used to obtain a CDT, the constrained edges simply are not tested because they can not be swapped.

3.4.2.5 Edge insertion:

This algorithm solves the problem of minimizing the maximum angle in time $O(n^2\log(n))$ exactly. Like the edge flipping algorithm, it starts from an arbitrary initial triangulation. It incidentally inserts candidate edges on a vertex of a worst triangle. The crossed edges are removed, and the remaining regions are re-triangulated. If the triangulation gets worse the added edge is rejected. This algorithm can also be used to find an interpolating surface with minimal slope in time $O(n^3)$ (Bern and Eppstein 1995).

3.4.2.6 Blue, red and green refinement

This is another popular method for mesh refinement. Basically, an element with an aspect ratio out of bound gets divided into two elements by inserting a new vertex on the midpoint of the longest edge. This is the so-called green refinement. The new vertex has to be connected on both sides of the edge with the opposing vertices of the neighbouring elements.
If one of the two resulting triangles still has a bad aspect ratio, the element gets divided again in the same manner resulting in three new elements. This is called the blue refinement. It is the goal of the blue and green refinement to halve the longest side of the original triangle and whereas degeneration of the mesh in repeated mesh refinement steps is prevented. Finally, the red refinement connects all midpoints of the original edges which leads to four new elements.

This page has been intentionally left blank.

3.5 Use of the grid generator of SMS 10

Obviously, the pre-processing before running a simulation needs a lot of work and effort. The input for the numerical module consists of a computational mesh, geometric and topographic attributes, boundary conditions, material properties (e.g. friction factor) etc. A suitable pre-processing unit should therefore satisfy as much as possible of the following requirements:

1) Grid generation
    - 1-1) Import of referenced Image data (TIFF) and DXF;
    - 1-2) Capability to produce triangular and quadrilateral grids;
    - 1-3) Mapping of point clouds on a computational grid by interpolation;
    - 1-4) Tools for refinement and coarsening of the mesh;
    - 1-5) Tools for control over the mesh quality;
    - 1-6) Enumeration tool;
    - 1-7) Possibility for geometric intervention;

2) Boundary Conditions
    - 2-1) Definition of inflow and outflow – stationary and instationary in time
    - 2-2) Definition of discharge directions
    - 2-3) Surfaces of different roughness

3) Computational methods
    - 3-1) stationary and instationary
    - 3-2) wet-dry criterion
    - 3-3) computational time

The development of a pre-processing module had no priority. Therefore, some commercial products have been evaluated on the mentioned criteria. The program SMS 10 (Surface Water Modelling System) proved to be best suited for all requirements.
SMS is an overall graphic user interface for mesh generation and manipulation – specialized on surface water flows. Furthermore, it is capable for post processing tasks like visualization of the computed results. The program is mainly used for the setup of the computational grid and the definition of boundary conditions.

SMS provides various modules for mesh generation. For the use with BASEMENT, we recommend the following modules (map, mesh and scatter modules are contained in the "SINGLE MODEL" package of SMS as generic model interface, price: ca. $2500).

(1) Mesh-module
The mesh module constructs a two-dimensional computational grid of rivers, bays and ports for which the shallow water equations are valid. A variety of sophisticated and automated tools allow fort the treatment of even difficult topographic situations.

(2) Scattered Data-Module
This module is used for interpolation of a group of widespread points into suitable meshes. Interpolation can be caused by data of vertices, measured field data or other sources delivered by the user. This feature is intended for the setup of initial conditions or the examination of the existing model. The program supports a broad range of different interpolation methods.

(3) Map -Module
GIS-objects are used within SMS to illustrate topographic attributes. SMS assigns properties and conditions to these GIS-objects and constructs a mesh of elements. This reduces the grid setup and development dramatically. SMS imports TIFF and DXF data and uses them to develop and alter specific areas and model boundaries.

SMS 10 was used and tested during the development of the BASEMENT software. After a year of experience, it has proved to satisfy the needs concerning functionality and suitability. The interactive interfaces provide a wide range of possibilities which simplify the process of grid adaptation and definition of the input data.
The visualization can be easily performed with SMS. Scalar values like water depth or shear stress are rendered as contour plots while vector quantities as velocities are visualized in a scaled manner. Another important feature is the possibility to produce animated graphics in time. Like this, the results can be visualized. Most of the results in manual RIII: Test Cases were produced using SMS 10

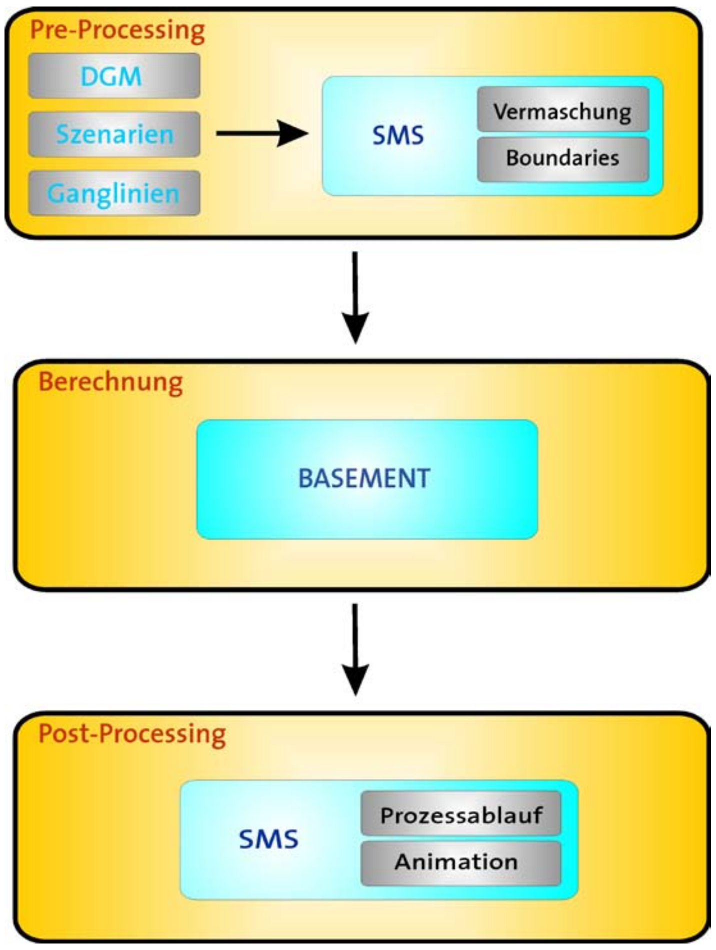


Fig. 9: Simulation procedure with use of SMS

.

This page has been intentionally left blank.

References

Bath, T. J. (1994). “Aspects of unstructured grids and finite –volume solvers for the Euler and Navier-Stokes quations”. *VKI Lecture Series* 1994-05.

Bern, M., and Eppstein, D. (1995). “Mesh generation and optimal triangulation”. *Computing in Euclidean Geometry*. D.Z. Du and F.K. Hwang, editors, 2nd Edition, pp. 47-123, World Scientific, Singapore, 1995.

Bern, M.,. and Plassman, P. (2000). “Mesh Generation”. *Handbook of Computational Geometry*, J. Sack and J. Urrutia, editors Elsevier Science, 2000.

This page has been intentionally left blank.

GRAPHICAL USER INTERFACE

of ***BASEMENT***

This page has been intentionally left blank.

Table of Contents

1 General

1.1 Purpose of this manual 1.1-1

2 The Graphical User Interface (GUI)

2.1 The BASEMENT Main Window 2.1-1
2.2 General Topics of Editing Files 2.2-1
2.2.1 BASEMENT Editor 2.2-1
2.2.2 Edit Blocks and Tags 2.2-1
2.2.2.1 Add Blocks 2.2-1
2.2.2.2 Define Tags 2.2-2
2.2.2.3 Remove Blocks or Tags 2.2-2
2.2.3 Automatic Validation 2.2-2
2.2.3.1 Mandatory Blocks and Tags 2.2-2
2.2.3.2 Default Values 2.2-2
2.2.4 Use the Built-in Help Function 2.2-2

3 Edit Command File

3.1 The “BASEMENT Command File Editor” 3.1-1
3.2 Create New or Edit Existing Command File 3.2-1
3.3 Tools 3.3-1
3.3.1 Edit Raw 3.3-1

4 Edit 1-D Grid

4.1 The “BASEMENT 1-D Grid File Editor” 4.1-1
4.2 Create New or Edit Existing Geometry File 4.2-1
4.3 Tools 4.3-1
4.3.1 Edit Raw 4.3-1
4.3.2 Friction 4.3-1
4.3.3 Remove Nodes 4.3-2
Guess Active Range 4.3-3
4.3.4 Guess Fixpoints 4.3-3
4.3.5 Interpolation 4.3-4
4.3.6 Export DTM for BASEplane 4.3-7

5 Built-In GUI Tools

5.1 Interactive Visualization during run time using BASEviz 5.1-1
5.2 Manual Controller Interface (HID) 5.2-1

List of Figures

Fig. 1: BASEMENT Main Window with menu bar, action buttons, splash screen and a progress bar.2.1-1

Fig. 2: Main Structure the BASEMENT Command File Editor..2.2-1

Fig. 3: BASEMENT Command File Editor..3.1-1

Fig. 4: Command File Editor: Raw Edit Window..3.3-1

Fig. 5: Grid File Editor – Subdomain View..4.1-1

Fig. 6: Grid File Editor – Cross Section View 4.2-1

Fig. 7: Grid File Editor: Raw Edit Window 4.3-1

Fig. 8: Friction assignment 4.3-2

Fig. 9: Node removal dialog for setting up the tolerance...4.3-2

Fig. 10: Schematic sketch of node removal 4.3-3

Fig. 11: in red the guessed fix points for cross section interpolation...4.3-4

Fig. 13: Curved alignment of the cross sections...4.3-5

Fig. 14: Setup dialog for the cross section interpolation...4.3-6

Fig. 15: GUI of the BASEMENT 1D grid file editor. The 1-D BASECHAIN_GEOMETRY can be exported with Export DTM for BASEplane under the menu bar Tools............................4.3-8

Fig. 16: Visualization of BASEchain with BASEviz..5.1-1

Fig. 17: Visualization of BASEplane with BASEviz...5.1-2

Fig. 18: Interface for the manual control and monitoring of the selected variables. 5.2-1

III GRAPHICAL USER INTERFACE

1 General

1.1 Purpose of this manual

Since Version 2.0, BASEMENT is not a pure console application anymore. The Graphic User Interface (GUI) enhances the usability but it is still possible to work with BASEMENT on a console.

It is now possible to define the whole Command File using the GUI. This avoids Syntax Errors within the Command File compared to the versions where the Command File was created manually. Furthermore, the *Command File Editor* validates the current state of the Input File and indicates whether there remain some Errors or Warnings.

Grid generation for 1-D simulations has never been easier. The *1-D Grid File Editor* allows the definition of cross sections and provides a graphical visualisation of the whole river reach and also every single cross section. There are many additional tools to work on cross sections which facilitate the setup of a 1-D geometry.

The intention of this document is to help the user working with the Graphic User Interface. The common basic behaviour is explained in Chapter 2. The Use of the Command File Editor is described in Chapter 3 and the Explanation of the 1-D Grid File Editor can be found in Chapter 4. In Chapter 5, interactive Visualisation and Manipulation tools during the runtime of a simulation are shown.

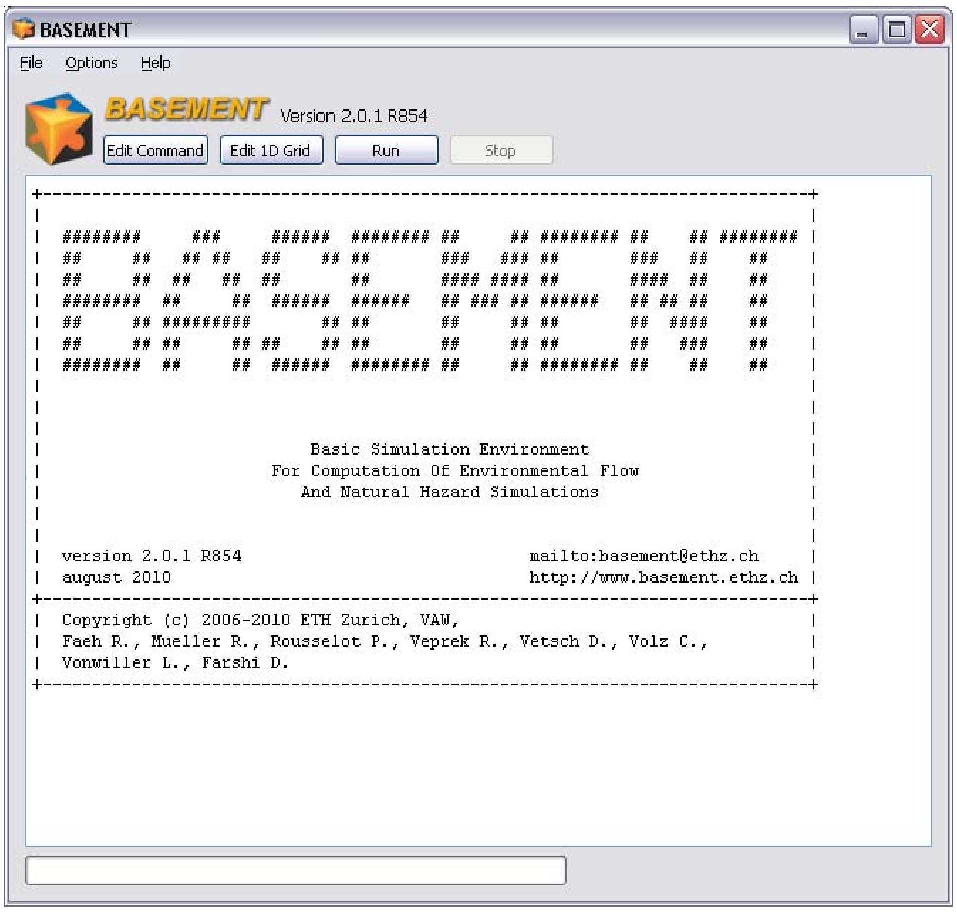
In some situations it may be desired to run the program without GUI and without user interaction, e.g. to execute several program runs via batch script over night or when running the program on a high-performance machine. Further details concerning the batch execution mode are explained in the user manual UI.

This page has been intentionally left blank.

2 The Graphical User Interface (GUI)

2.1 The BASEMENT Main Window

Starting BASEMENT will first produce a Graphical User Interface as shown in *Fig. 1*. On the top of the screen, a menu bar consisting of the menus *File*, *Options* and *Help* is available. Below the menu bar, near the BASEMENT logo, there are four action buttons named *Edit Command*, *Edit 1D Grid*, *Run* and *Stop*. The main window displays a splash screen indicating the current version of the Software. A yet empty progress bar is located below the main window.



```
+------------------------------------------------------------------------------+
|                                                                              |
|  ########     ###     ######  ######## ##     ## ######## ##    ## ########  |
|  ##     ##   ## ##   ##    ## ##       ###   ### ##       ###   ##    ##     |
|  ##     ##  ##   ##  ##       ##       #### #### ##       ####  ##    ##     |
|  ########  ##     ##  ######  ######   ## ### ## ######   ## ## ##    ##     |
|  ##     ## #########       ## ##       ##     ## ##       ##  ####    ##     |
|  ##     ## ##     ## ##    ## ##       ##     ## ##       ##   ###    ##     |
|  ########  ##     ##  ######  ######## ##     ## ######## ##    ##    ##     |
|                                                                              |
|                                                                              |
|                                                                              |
|                           Basic Simulation Environment                       |
|                       For Computation Of Environmental Flow                  |
|                          And Natural Hazard Simulations                      |
|                                                                              |
|                                                                              |
|  version 2.0.1 R854                             mailto:basement@ethz.ch      |
|  august 2010                                    http://www.basement.ethz.ch  |
+------------------------------------------------------------------------------+
|  Copyright (c) 2006-2010 ETH Zurich, VAW,                                    |
|  Faeh R., Mueller R., Rousselot P., Veprek R., Vetsch D., Volz C.,           |
|  Vonwiller L., Farshi D.                                                     |
+------------------------------------------------------------------------------+
```

Fig. 1: BASEMENT Main Window with menu bar, action buttons, splash screen and a progress bar.

The menu *File* allows for opening either an existing command file (*Open Command*) or an existing 1-D Grid (*Open 1D Grid*) within the BASEMENT Editor. Selecting *Quit* from the *File* menu will exit the Application.

The menu *Options* hosts the setting for the *Log level*. Different Log levels are available. A Log level 5 will show all possible Log output, whereas Log level 0 will suppress most of the Log information from the application.

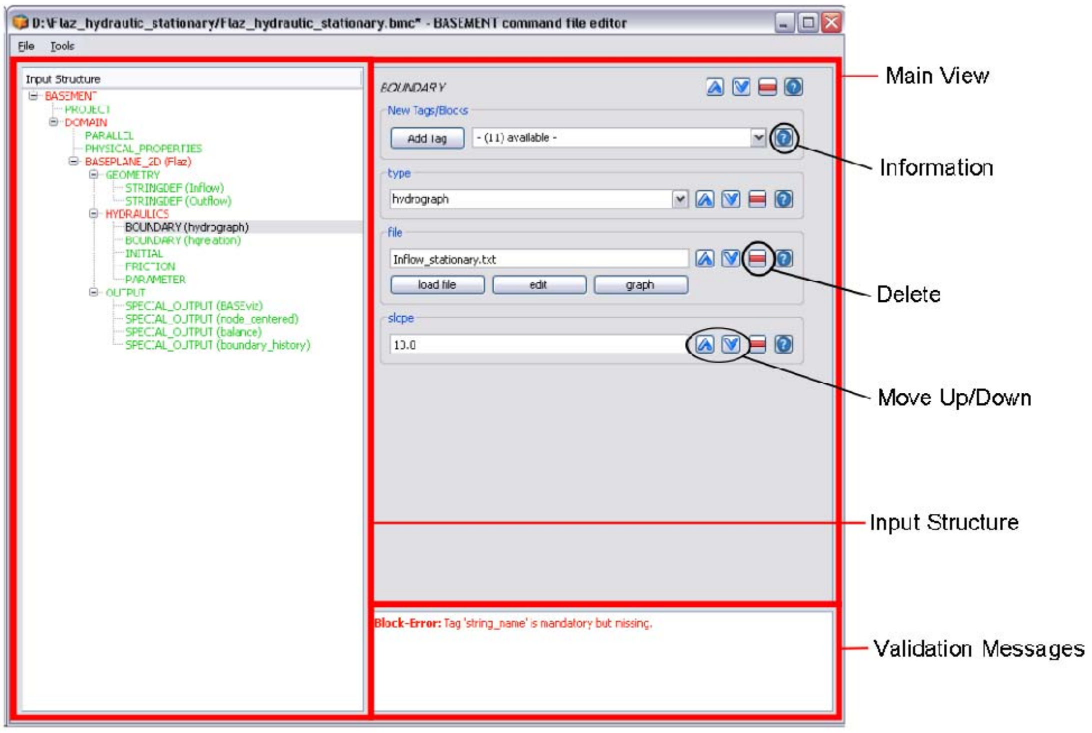
The splash screen can be shown anytime again by selecting *About BASEMENT* from the *Help* menu.
The action buttons have the following behavior:

- *Edit Command* opens the currently active Command File within the Command File Editor. If no Command File has been opened yet, the Editor will be empty. A description of the Command File Editor is given in chapter 3.
- *Edit 1D Grid* opens the currently active 1D Grid definition within the 1-D Grid File Editor. If no 1-D Grid has been opened yet, the Editor will be empty. A description of the 1-D Grid File Editor is given in chapter 0.
- *Run* is only active when a Command File is active in the background. Pressing this button will start a simulation according to the settings in the Command File.
- *Stop* is only active if a simulation is currently being carried out. Pressing this button will stop the current simulation.

2.2 General Topics of Editing Files

2.2.1 BASEMENT Editor

Both, the Command File Editor and the 1-D Grid File Editor share a common composition. The Window is subdivided into three parts: the Input Structure to the left, the Main View to the right and the Validation Messages at the lower right corner (*Fig. 2*).
The Input Structure shows the tree of all Blocks defined and allows the selection of a specific Block. The Main View shows all relevant Information for the currently selected Block like available Sub-Blocks/Tags and the values for all chosen Tags. Graphical visualization of e.g. a Cross-section or a time series can also be seen.
The Validation Messages indicate by their Color whether the Input for the selected Block contains any Errors or Warnings.


D:\Flaz_hydraulic_stationary/Flaz_hydraulic_stationary.bmc* - BASEMENT command file editor

```
Input Structure
⊟ BASEMENT
   PROJECT
   ⊟ DOMAIN
      PARALLEL
      PHYSICAL_PROPERTIES
      ⊟ BASEPLANE_2D (Flaz)
         ⊟ GEOMETRY
            STRINGDEF (Inflow)
            STRINGDEF (Outflow)
         ⊟ HYDRAULICS
            BOUNDARY (hydrograph)
            BOUNDARY (hqrelation)
            INITIAL
            FRICTION
            PARAMETER
         ⊟ OUTPUT
            SPECIAL_OUTPUT (BASEviz)
            SPECIAL_OUTPUT (node_centered)
            SPECIAL_OUTPUT (balance)
            SPECIAL_OUTPUT (boundary_history)
```

BOUNDARY

New Tags/Blocks: - (11) available -

type: hydrograph

file: Inflow_stationary.txt

slope: 10.0

Block-Error: Tag 'string_name' is mandatory but missing.

Fig. 2: Main Structure the BASEMENT Command File Editor.

2.2.2 Edit Blocks and Tags

2.2.2.1 Add Blocks

Select a Block from the Input Structure. Use the drop down menu near the *Add Block* button in the Main View to see all available Blocks. Select one of them and press the *Add Block* button to add a new Block.

2.2.2.2 Define Tags

Select a Block from the Input Structure. Use the drop down menu near the *Add Tag* button in the main view to see all available Tags. Select one of them and press the *Add Tag* button to add a new tag. The Tag will appear in the main view and needs to be given a precise value. Depending on the Tag, a value can be chosen from a drop down menu or has to be defined by the user.

2.2.2.3 Remove Blocks or Tags

To remove a Block or a Tag, use the Delete button as indicated in *Fig. 2*. For a block, this is located at the top of the main view near the Blocks name. For a Tag, the button is to the right of the precise value of a Tag. There will be a confirmation message whether a Block or Tag really shall be removed.

2.2.3 Automatic Validation

Validation Messages indicate immediately whether the current choice of Blocks and Tags is valid (*Fig. 2*). In case of errors, the Block and its parent Blocks in the Input Structure are red. In case of warnings, the Block and its parent Blocks in the Input Structure are brown. Blocks without warnings or errors are green.
Selecting a red or brown Block in the Input Structure will show all errors and warnings as Validation Messages.
An Input without red messages has no errors and can be used for simulation.

2.2.3.1 Mandatory Blocks and Tags

Certain Blocks and Tags may be mandatory to define. If a mandatory Block or Tag is missing, an error message like *Block Error: Tag "XY" is mandatory but missing* is displayed until the Block or Tag is defined.

2.2.3.2 Default Values

Some Tags have default values. They are usually not mandatory to define. Defining a Tag with an associated default value will show the specific Tag value in the Input Field for the Tag where it can also be changed to a user defined value.

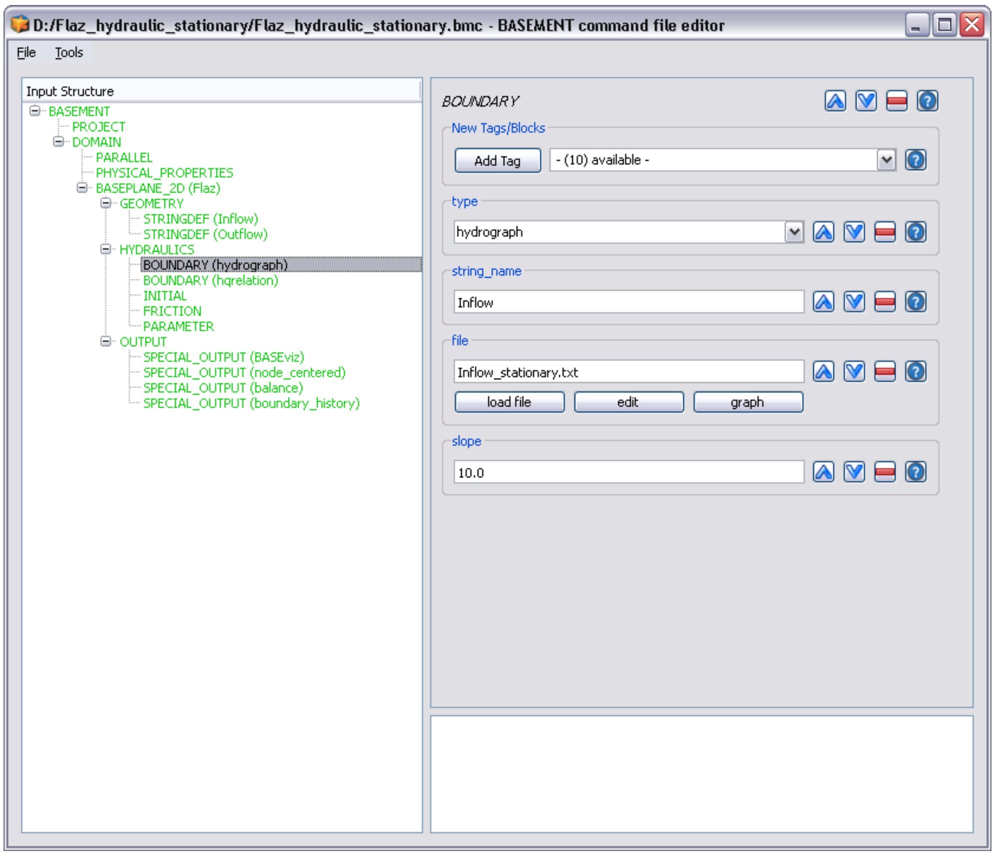
2.2.4 Use the Built-in Help Function

Every Block and Tag is documented within the Reference Manual R IV. This same information is also available within the BASEMENT Editor. Clicking on the Information button with a question mark (*Fig. 2*) will produce a pop up window with a description of the usage and some examples. The Information button for a block is located at the top of the main view near the Blocks name. For a Tag, the button is to the right of the precise value of a Tag. The documentation for Tags also includes information about whether it is mandatory, default values, value range, etc.

3 Edit Command File

3.1 The “BASEMENT Command File Editor”

From the BASEMENT main window (*Fig. 1*) the user gets via the *Edit Command* button to the *Command File Editor* which is shown in *Fig. 3*. The Input Structure (as it is built up with Blocks) is on the left-hand side. The Main View on the right-hand side gives more details of the selected Block with the corresponding Tags and the possibility to add Tags and/or Blocks (see chapter 2).


D:/Flaz_hydraulic_stationary/Flaz_hydraulic_stationary.bmc - BASEMENT command file editor

File Tools

Input Structure

```
BASEMENT
  PROJECT
  DOMAIN
    PARALLEL
    PHYSICAL_PROPERTIES
    BASEPLANE_2D (Flaz)
      GEOMETRY
        STRINGDEF (Inflow)
        STRINGDEF (Outflow)
      HYDRAULICS
        BOUNDARY (hydrograph)
        BOUNDARY (hqrelation)
        INITIAL
        FRICTION
        PARAMETER
      OUTPUT
        SPECIAL_OUTPUT (BASEviz)
        SPECIAL_OUTPUT (node_centered)
        SPECIAL_OUTPUT (balance)
        SPECIAL_OUTPUT (boundary_history)
```

BOUNDARY

New Tags/Blocks: Add Tag | - (10) available -

type: hydrograph

string_name: Inflow

file: Inflow_stationary.txt

load file | edit | graph

slope: 10.0

Fig. 3: BASEMENT Command File Editor

This page has been intentionally left blank.

3.2 Create New or Edit Existing Command File

A new Command File is created form the menu bar in the *Command File Editor*. Choose *File* on the menu bar and select *New*. The whole command file consisting of Blocks and Tags can be built up from scratch. For further details on how to add Blocks and define Tags see chapter 2.2.2 in this manual.

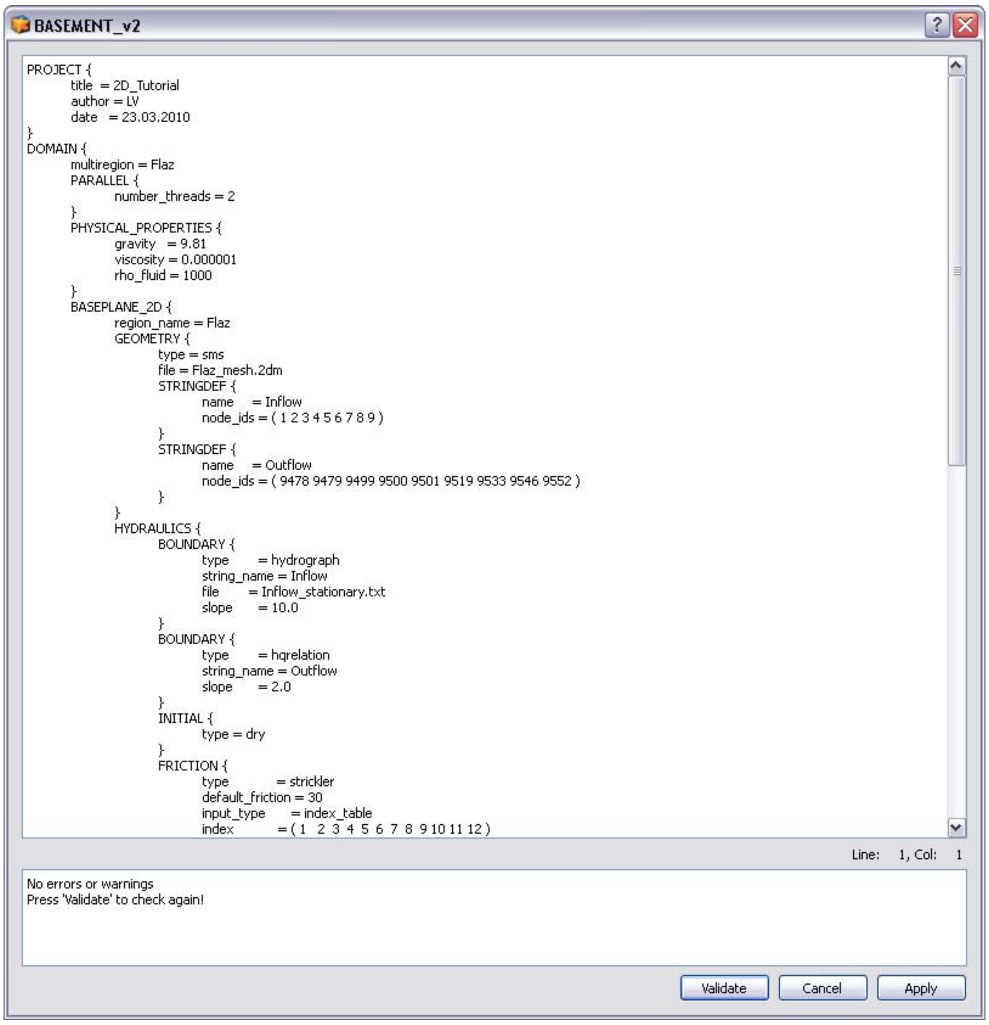
If you want to open an existing command file you can do this from the *Command File Editor* or directly from the *BASEMENT Main Window*. From the *Command File Editor* choose *File* on the menu bar and then select *Open*. Then you browse and choose your file as usually. Don’t forget to save your changes made in the command file before you run a simulation.

This page has been intentionally left blank.

3.3 Tools

3.3.1 Edit Raw

Another way to edit a command file is to edit basically the command file in a raw text mode. For this purpose choose *Tools* on the menu bar and select *Edit Raw*. The Editor window will pop up as shown in *Fig. 4*. Long-standing users of BASEMENT will recognize the input structure of the former command file. In the lower part of the window the Input is validated and possible parse errors are indicated. For the sake of completeness it is mentioned here that the command file can still be built up and edited with a simple text editor.



```
PROJECT {
        title  = 2D_Tutorial
        author = LV
        date   = 23.03.2010
}
DOMAIN {
        multiregion = Flaz
        PARALLEL {
                number_threads = 2
        }
        PHYSICAL_PROPERTIES {
                gravity   = 9.81
                viscosity = 0.000001
                rho_fluid = 1000
        }
        BASEPLANE_2D {
                region_name = Flaz
                GEOMETRY {
                        type = sms
                        file = Flaz_mesh.2dm
                        STRINGDEF {
                                name     = Inflow
                                node_ids = ( 1 2 3 4 5 6 7 8 9 )
                        }
                        STRINGDEF {
                                name     = Outflow
                                node_ids = ( 9478 9479 9499 9500 9501 9519 9533 9546 9552 )
                        }
                }
                HYDRAULICS {
                        BOUNDARY {
                                type        = hydrograph
                                string_name = Inflow
                                file        = Inflow_stationary.txt
                                slope       = 10.0
                        }
                        BOUNDARY {
                                type        = hqrelation
                                string_name = Outflow
                                slope       = 2.0
                        }
                        INITIAL {
                                type = dry
                        }
                        FRICTION {
                                type             = strickler
                                default_friction = 30
                                input_type       = index_table
                                index            = ( 1  2  3  4  5  6  7  8  9 10 11 12 )
```

```
No errors or warnings
Press 'Validate' to check again!
```

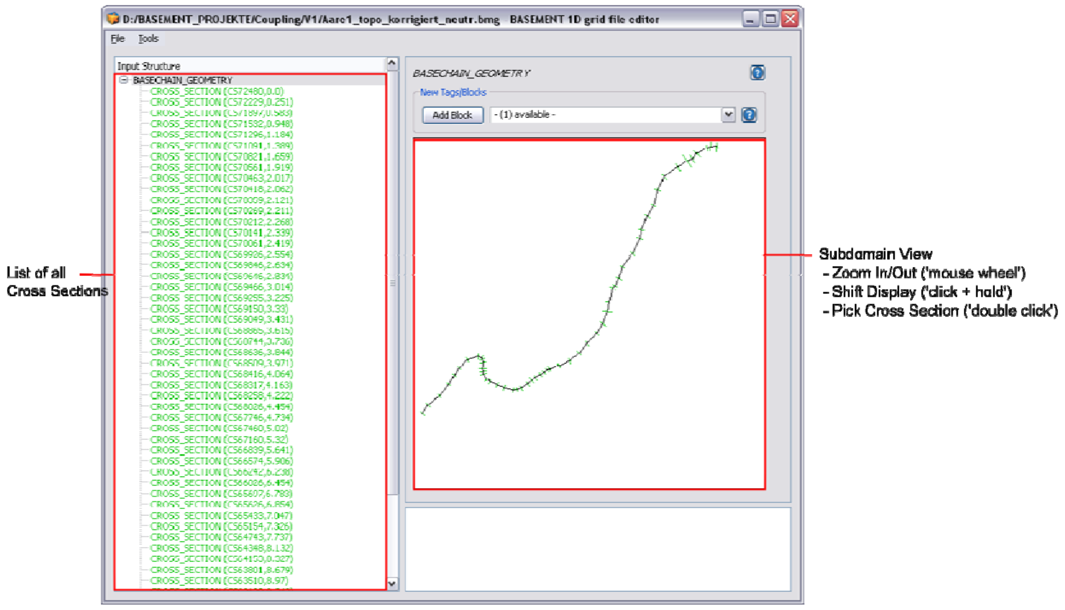
Fig. 4: Command File Editor: Raw Edit Window.

This page has been intentionally left blank.

4 Edit 1-D Grid

4.1 The “BASEMENT 1-D Grid File Editor”

When the Grid File Editor is opened a new window pops up (see *Fig. 5*). In the left part of the window a list of all cross sections is shown where the cross sections can be seen and selected. In the right part of the window a visualization of the whole subdomain with all cross sections is drawn and new cross sections can be created with the *Add Block* option. The subdomain view allows zooming and shifting of the display and the selection of a specific cross section by double clicking. If a cross section is selected then the view changes to the cross section view.


D:/BASEMENT_PROJEKTE/Coupling/V1/Aare1_topo_korrigiert_neutr.bmg BASEMENT 1D grid file editor
File
Tools
Input Structure
BASECHAIN_GEOMETRY
New Tags/Blocks
Add Block
- (1) available -
List of all Cross Sections
Subdomain View
- Zoom In/Out ('mouse wheel')
- Shift Display ('click + hold')
- Pick Cross Section ('double click')

```
BASECHAIN_GEOMETRY
    CROSS_SECTION (CS72480,0.0)
    CROSS_SECTION (CS72229,0.251)
    CROSS_SECTION (CS71897,0.583)
    CROSS_SECTION (CS71532,0.948)
    CROSS_SECTION (CS71296,1.184)
    CROSS_SECTION (CS71091,1.389)
    CROSS_SECTION (CS70821,1.659)
    CROSS_SECTION (CS70561,1.919)
    CROSS_SECTION (CS70463,2.017)
    CROSS_SECTION (CS70418,2.062)
    CROSS_SECTION (CS70009,2.121)
    CROSS_SECTION (CS70289,2.211)
    CROSS_SECTION (CS70212,2.268)
    CROSS_SECTION (CS70141,2.339)
    CROSS_SECTION (CS70061,2.419)
    CROSS_SECTION (CS69926,2.554)
    CROSS_SECTION (CS69846,2.634)
    CROSS_SECTION (CS69646,2.834)
    CROSS_SECTION (CS69466,3.014)
    CROSS_SECTION (CS69255,3.225)
    CROSS_SECTION (CS69150,3.33)
    CROSS_SECTION (CS69049,3.431)
    CROSS_SECTION (CS68865,3.615)
    CROSS_SECTION (CS68744,3.736)
    CROSS_SECTION (CS68636,3.844)
    CROSS_SECTION (CS68509,3.971)
    CROSS_SECTION (CS68416,4.064)
    CROSS_SECTION (CS68317,4.163)
    CROSS_SECTION (CS68258,4.222)
    CROSS_SECTION (CS68026,4.454)
    CROSS_SECTION (CS67746,4.734)
    CROSS_SECTION (CS67460,5.02)
    CROSS_SECTION (CS67160,5.32)
    CROSS_SECTION (CS66839,5.641)
    CROSS_SECTION (CS66574,5.906)
    CROSS_SECTION (CS66292,6.238)
    CROSS_SECTION (CS66086,6.454)
    CROSS_SECTION (CS65607,6.783)
    CROSS_SECTION (CS65526,6.864)
    CROSS_SECTION (CS65433,7.047)
    CROSS_SECTION (CS65154,7.326)
    CROSS_SECTION (CS64743,7.737)
    CROSS_SECTION (CS64348,8.132)
    CROSS_SECTION (CS64153,8.327)
    CROSS_SECTION (CS63801,8.679)
    CROSS_SECTION (CS63510,8.97)
```

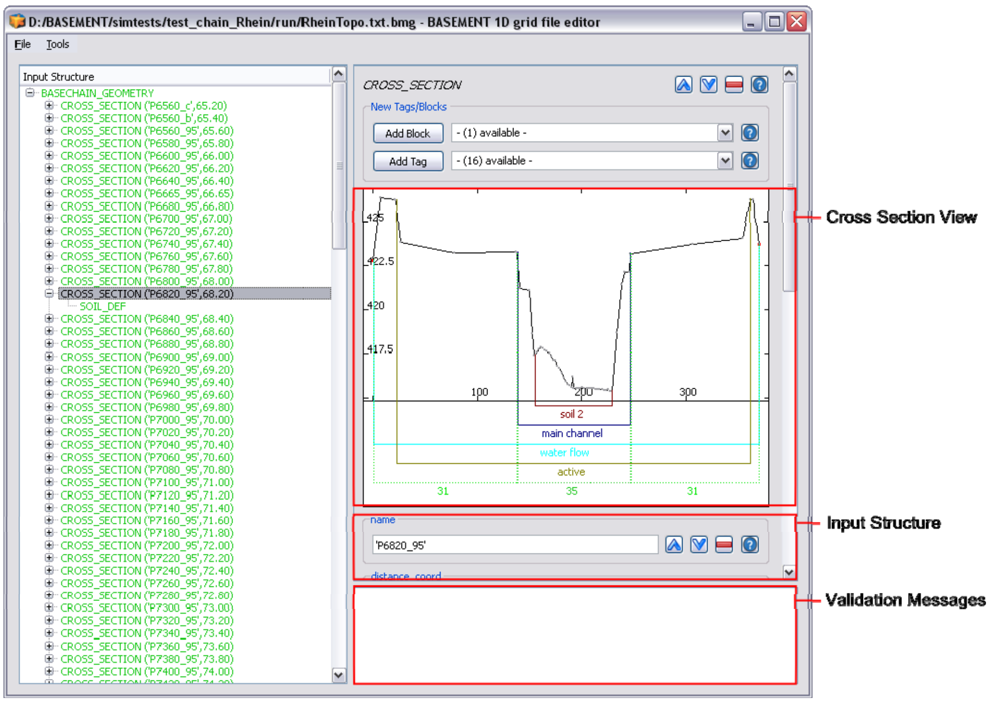
Fig. 5: Grid File Editor – Subdomain View

The real (usually curved) shape of the stream can only be illustrated if all cross sections are geo-referenced and if all corresponding data is set (the *orientation_angle* and the *left_point_global_coordinates* must be set for each cross section). If these data are not given than the cross sections are drawn along a straight line.

This page has been intentionally left blank.

4.2 Create New or Edit Existing Geometry File

If a specific cross section is selected or a new cross section is created, than a profile view of the selected cross section is shown (see *Fig. 6*). With this visualization of the profile one can easily check for input errors in the geometrical definition of the cross section profile. Furthermore, the most important cross section parameters are indicated visually with different colours, like e.g. the definition of the main channel, the range of the soils, the friction parameters, the cross section fixpoints, etc. Again, one can visually check if these parameters are set up correctly and thus easily detect type errors. New input tags can be added and the validation message box shows warnings or errors if some problematic inputs have been made.


D:/BASEMENT/simtests/test_chain_Rhein/run/RheinTopo.txt.bmg - BASEMENT 1D grid file editor

```
Input Structure
⊟ BASECHAIN_GEOMETRY
    ⊞ CROSS_SECTION ('P6560_c',65.20)
    ⊞ CROSS_SECTION ('P6560_b',65.40)
    ⊞ CROSS_SECTION ('P6560_95',65.60)
    ⊞ CROSS_SECTION ('P6580_95',65.80)
    ⊞ CROSS_SECTION ('P6600_95',66.00)
    ⊞ CROSS_SECTION ('P6620_95',66.20)
    ⊞ CROSS_SECTION ('P6640_95',66.40)
    ⊞ CROSS_SECTION ('P6665_95',66.65)
    ⊞ CROSS_SECTION ('P6680_95',66.80)
    ⊞ CROSS_SECTION ('P6700_95',67.00)
    ⊞ CROSS_SECTION ('P6720_95',67.20)
    ⊞ CROSS_SECTION ('P6740_95',67.40)
    ⊞ CROSS_SECTION ('P6760_95',67.60)
    ⊞ CROSS_SECTION ('P6780_95',67.80)
    ⊞ CROSS_SECTION ('P6800_95',68.00)
    ⊟ CROSS_SECTION ('P6820_95',68.20)
        SOIL_DEF
    ⊞ CROSS_SECTION ('P6840_95',68.40)
    ⊞ CROSS_SECTION ('P6860_95',68.60)
    ⊞ CROSS_SECTION ('P6880_95',68.80)
    ⊞ CROSS_SECTION ('P6900_95',69.00)
    ⊞ CROSS_SECTION ('P6920_95',69.20)
    ⊞ CROSS_SECTION ('P6940_95',69.40)
    ⊞ CROSS_SECTION ('P6960_95',69.60)
    ⊞ CROSS_SECTION ('P6980_95',69.80)
    ⊞ CROSS_SECTION ('P7000_95',70.00)
    ⊞ CROSS_SECTION ('P7020_95',70.20)
    ⊞ CROSS_SECTION ('P7040_95',70.40)
    ⊞ CROSS_SECTION ('P7060_95',70.60)
    ⊞ CROSS_SECTION ('P7080_95',70.80)
    ⊞ CROSS_SECTION ('P7100_95',71.00)
    ⊞ CROSS_SECTION ('P7120_95',71.20)
    ⊞ CROSS_SECTION ('P7140_95',71.40)
    ⊞ CROSS_SECTION ('P7160_95',71.60)
    ⊞ CROSS_SECTION ('P7180_95',71.80)
    ⊞ CROSS_SECTION ('P7200_95',72.00)
    ⊞ CROSS_SECTION ('P7220_95',72.20)
    ⊞ CROSS_SECTION ('P7240_95',72.40)
    ⊞ CROSS_SECTION ('P7260_95',72.60)
    ⊞ CROSS_SECTION ('P7280_95',72.80)
    ⊞ CROSS_SECTION ('P7300_95',73.00)
    ⊞ CROSS_SECTION ('P7320_95',73.20)
    ⊞ CROSS_SECTION ('P7340_95',73.40)
    ⊞ CROSS_SECTION ('P7360_95',73.60)
    ⊞ CROSS_SECTION ('P7380_95',73.80)
    ⊞ CROSS_SECTION ('P7400_95',74.00)
```

CROSS_SECTION

New Tags/Blocks

Add Block: - (1) available -

Add Tag: - (16) available -

name: 'P6820_95'

Fig. 6: Grid File Editor – Cross Section View

For details on how to set up a new cross section and for information about the various cross section parameters see the 1-D tutorial in the manual UIV.

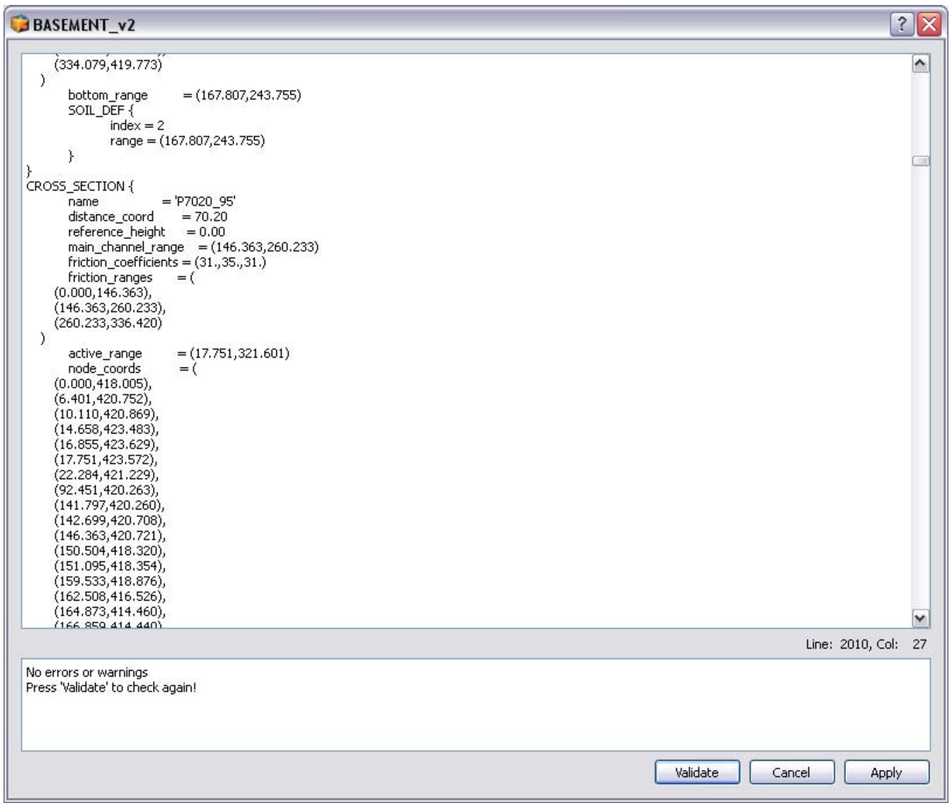
This page has been intentionally left blank.

4.3 Tools

BASEMENT supports several tools which support the user in creating and modifying the 1D grid file. In the following sections some information about the usage and the methods of these tools are given. Please be aware that some of the offered tools are still in a beta-status.

4.3.1 Edit Raw

Another way to edit a grid file is to edit the grid file in raw text mode. For this purpose choose *Tools* on the menu bar and select *Edit Raw*. The Editor window will pop up as shown in *Fig. 7****Fig. 4***. In the lower part of the window the Input is validated and possible parse errors are indicated. For the sake of completeness it is mentioned here that the grid file can still be built up and edited with a simple text editor.



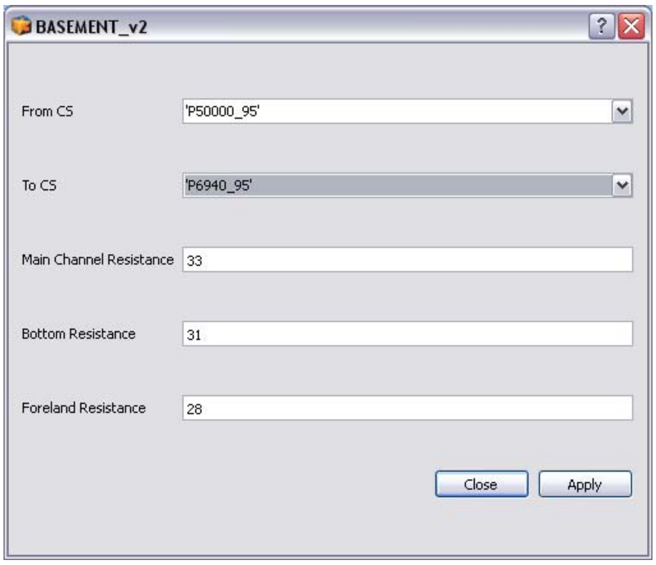
```
    (334.079,419.773)
  )
        bottom_range        = (167.807,243.755)
        SOIL_DEF {
              index = 2
              range = (167.807,243.755)
        }
}
CROSS_SECTION {
        name              = 'P7020_95'
        distance_coord      = 70.20
        reference_height     = 0.00
        main_channel_range    = (146.363,260.233)
        friction_coefficients = (31.,35.,31.)
        friction_ranges      = (
    (0.000,146.363),
    (146.363,260.233),
    (260.233,336.420)
  )
        active_range        = (17.751,321.601)
        node_coords         = (
    (0.000,418.005),
    (6.401,420.752),
    (10.110,420.869),
    (14.658,423.483),
    (16.855,423.629),
    (17.751,423.572),
    (22.284,421.229),
    (92.451,420.263),
    (141.797,420.260),
    (142.699,420.708),
    (146.363,420.721),
    (150.504,418.320),
    (151.095,418.354),
    (159.533,418.876),
    (162.508,416.526),
    (164.873,414.460),
```

```
No errors or warnings
Press 'Validate' to check again!
```

Fig. 7: Grid File Editor: Raw Edit Window

4.3.2 Friction

With this tool the friction value of the main channel, the forelands and the bottom can be assigned to a range of consecutive cross sections. The names of the first and the last cross section of the range have to be given.


BASEMENT_v2

From CS 'P50000_95'

To CS 'P6940_95'

Main Channel Resistance 33

Bottom Resistance 31

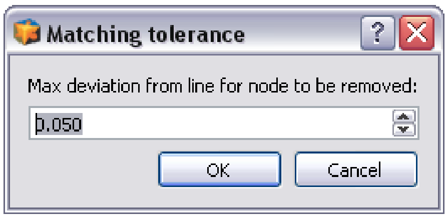
Foreland Resistance 28

Fig. 8: Friction assignment

4.3.3 Remove Nodes

Cross sections often consist of a large number of nodes and slices which consumes significant computational performance. If multiple nodes lie on a straight line within the cross section there is redundant information present which can be removed without reducing the accuracy of the computations. Therefore identifying and removing these redundant nodes is a frequent and recommended task before running the simulations. BASEMENT offers a tool which performs this task automatically without need for costly manual operations.

To access this tool open the *Tools* menu and click on the *Remove nodes* option. The following dialog opens:


Matching tolerance

Max deviation from line for node to be removed:

0.050

Fig. 9: Node removal dialog for setting up the tolerance

In this dialog you can define the maximum tolerance by which a node may deviate from the straight line of its two neighbours. If the node lies within this tolerance the node removal is applied. If you set the tolerance to small values only nodes are removed which are almost exactly situated on the line. If you increase the tolerance more nodes will be removed but some more information about the cross section profile may get lost. Usually one should try different tolerances until the best compromise between computational performance and accuracy is found. Also, the algorithm can be applied multiple times.

The applied algorithm loops all nodes of the cross sections and checks if a node is situated on a straight line between its two neighboured nodes (considering the given tolerance). If this is the case the node is removed from the profile (see *Fig. 10*). Nodes which are used as fixpoint or which are explicitly referenced by a slice_index range are excluded from the algorithm and cannot be removed automatically.

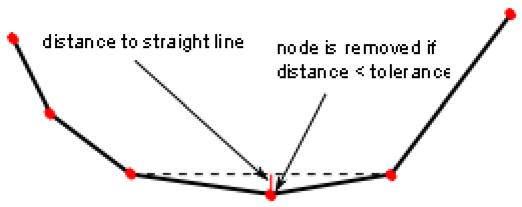


Fig. 10: Schematic sketch of node removal

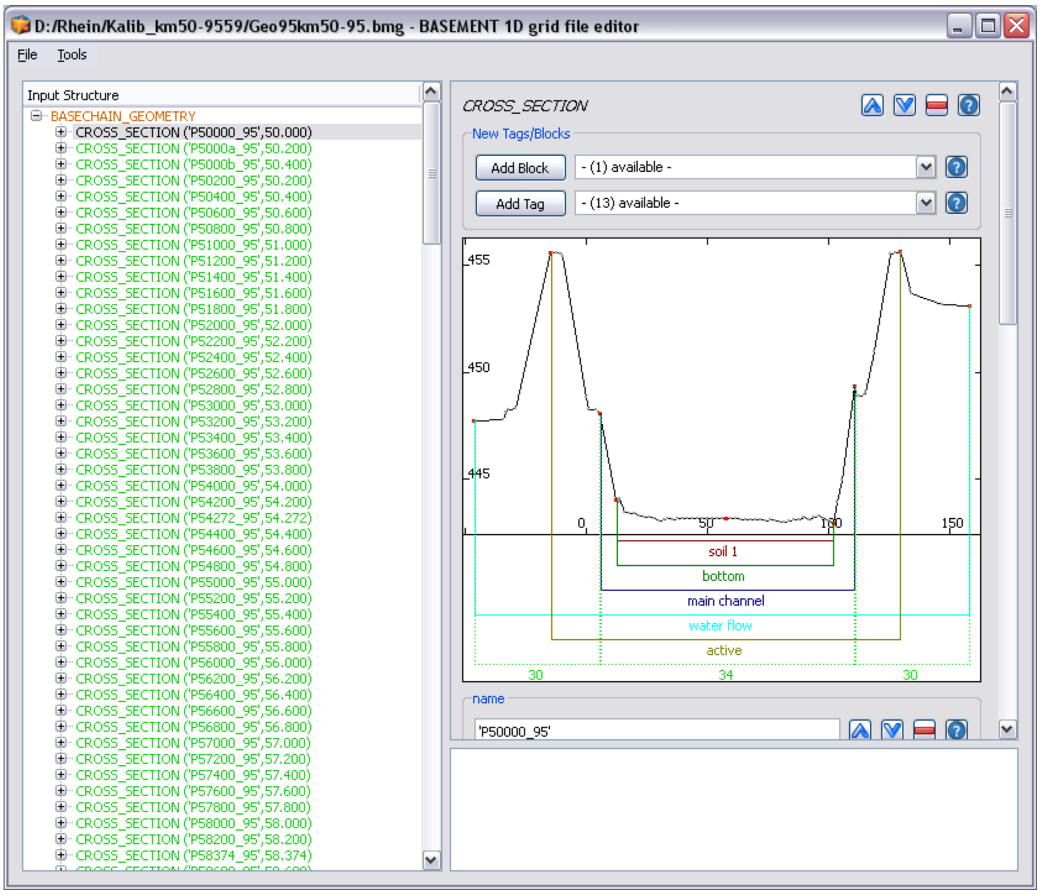
Guess Active Range

This tool provides a definition of the active range where it is not yet defined. For this purpose the lowest point of the cross section is searched and then the highest points to the left and the right of it (usually the dam crests) are set as limits of the active range. The definition of the active range can always be changed manually by the user in the interface. As an example on how the active range is set see *Fig. 11*.

4.3.4 Guess Fixpoints

This tool provides the definition of some fix points, which are needed for a correct interpolation of new cross sections between existing cross sections. So this should be done before an interpolation is executed. The fix points are displayed in red. The points which are automatically set as fix points are:

- The limits of soils
- The limits of the main channel
- The limits of the active range
- The midpoint of the main channel.


D:/Rhein/Kalib_km50-9559/Geo95km50-95.bmg - BASEMENT 1D grid file editor

```
Input Structure
⊟ BASECHAIN_GEOMETRY
    ⊞ CROSS_SECTION ('P50000_95',50.000)
    ⊞ CROSS_SECTION ('P5000a_95',50.200)
    ⊞ CROSS_SECTION ('P5000b_95',50.400)
    ⊞ CROSS_SECTION ('P50200_95',50.200)
    ⊞ CROSS_SECTION ('P50400_95',50.400)
    ⊞ CROSS_SECTION ('P50600_95',50.600)
    ⊞ CROSS_SECTION ('P50800_95',50.800)
    ⊞ CROSS_SECTION ('P51000_95',51.000)
    ⊞ CROSS_SECTION ('P51200_95',51.200)
    ⊞ CROSS_SECTION ('P51400_95',51.400)
    ⊞ CROSS_SECTION ('P51600_95',51.600)
    ⊞ CROSS_SECTION ('P51800_95',51.800)
    ⊞ CROSS_SECTION ('P52000_95',52.000)
    ⊞ CROSS_SECTION ('P52200_95',52.200)
    ⊞ CROSS_SECTION ('P52400_95',52.400)
    ⊞ CROSS_SECTION ('P52600_95',52.600)
    ⊞ CROSS_SECTION ('P52800_95',52.800)
    ⊞ CROSS_SECTION ('P53000_95',53.000)
    ⊞ CROSS_SECTION ('P53200_95',53.200)
    ⊞ CROSS_SECTION ('P53400_95',53.400)
    ⊞ CROSS_SECTION ('P53600_95',53.600)
    ⊞ CROSS_SECTION ('P53800_95',53.800)
    ⊞ CROSS_SECTION ('P54000_95',54.000)
    ⊞ CROSS_SECTION ('P54200_95',54.200)
    ⊞ CROSS_SECTION ('P54272_95',54.272)
    ⊞ CROSS_SECTION ('P54400_95',54.400)
    ⊞ CROSS_SECTION ('P54600_95',54.600)
    ⊞ CROSS_SECTION ('P54800_95',54.800)
    ⊞ CROSS_SECTION ('P55000_95',55.000)
    ⊞ CROSS_SECTION ('P55200_95',55.200)
    ⊞ CROSS_SECTION ('P55400_95',55.400)
    ⊞ CROSS_SECTION ('P55600_95',55.600)
    ⊞ CROSS_SECTION ('P55800_95',55.800)
    ⊞ CROSS_SECTION ('P56000_95',56.000)
    ⊞ CROSS_SECTION ('P56200_95',56.200)
    ⊞ CROSS_SECTION ('P56400_95',56.400)
    ⊞ CROSS_SECTION ('P56600_95',56.600)
    ⊞ CROSS_SECTION ('P56800_95',56.800)
    ⊞ CROSS_SECTION ('P57000_95',57.000)
    ⊞ CROSS_SECTION ('P57200_95',57.200)
    ⊞ CROSS_SECTION ('P57400_95',57.400)
    ⊞ CROSS_SECTION ('P57600_95',57.600)
    ⊞ CROSS_SECTION ('P57800_95',57.800)
    ⊞ CROSS_SECTION ('P58000_95',58.000)
    ⊞ CROSS_SECTION ('P58200_95',58.200)
    ⊞ CROSS_SECTION ('P58374_95',58.374)
```

CROSS_SECTION

New Tags/Blocks

Add Block - (1) available -

Add Tag - (13) available -

name: 'P50000_95'

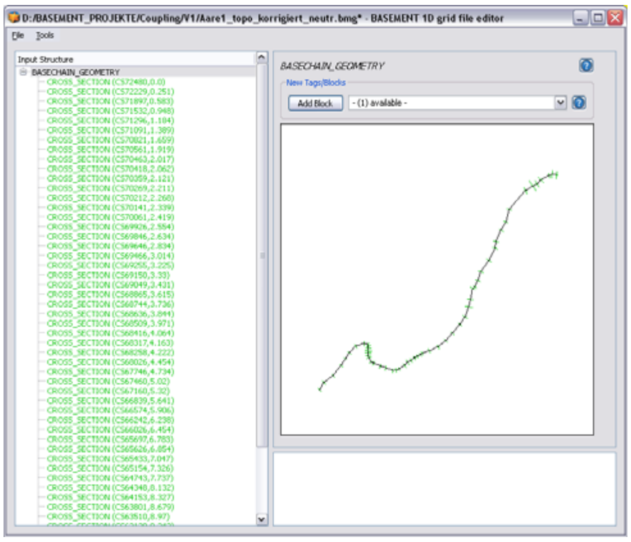
Fig. 11: in red the guessed fix points for cross section interpolation

It is recommended to check the points visually and add other important points, especially on the break lines. For the interpolation all involved cross sections must have the same number of fix points.

4.3.5 Interpolation

First of all, before an interpolation of 1-D cross sections can be performed some information is needed about the spatial alignment of all cross sections in the x-y plane.

There are two main tags which determine the spatial orientation of the cross section which are crucial for the interpolation algorithm. The *orientation_angle* provides the information which is needed for the orientation of the cross sections. This is the angle between the normal vector of the cross section and the vector in x-direction (1,0). Consequently, in a fully straight channel in x-direction all cross sections would have the angle 0° If the orientation angle is not given it is set to this value. The other essential tag is the *left_point_global_coordinates*. This parameter sets the global (real world) x,y,z-coordinates of the outer left point of the cross section. This parameter in combination with the orientation angle delivers all needed information about the spatial configuration of the cross section. If this coordinates are not given the value of *distance_coord* is used for x and the elevation of the first point on the left for z. y is set to negative distance of the first point on the left from the middle of the cross section. If both parameters are set for all cross sections, one can see the curved alignment of the stream in the right-hand visualization (see *Fig. 12*).



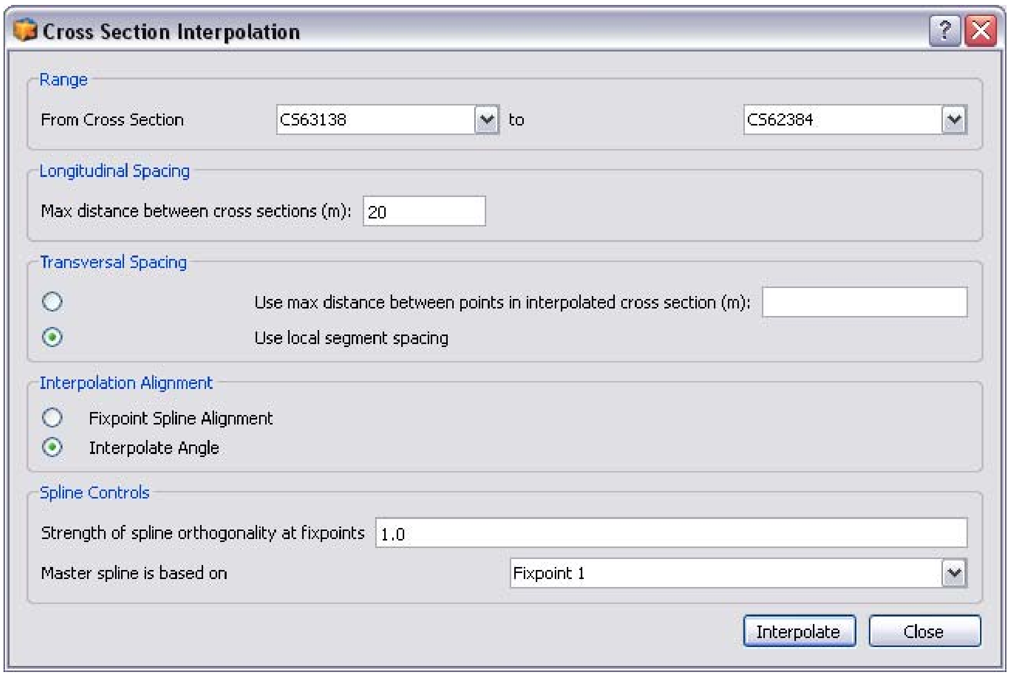
```
D:/BASEMENT_PROJEKTE/Coupling/V1/Aare1_topo_korrigiert_neutr.bmg* - BASEMENT 1D grid file editor
File  Tools

Input Structure
⊟ BASECHAIN_GEOMETRY
   CROSS_SECTION (CS72480,0.0)
   CROSS_SECTION (CS72229,0.251)
   CROSS_SECTION (CS71897,0.583)
   CROSS_SECTION (CS71532,0.948)
   CROSS_SECTION (CS71296,1.184)
   CROSS_SECTION (CS71091,1.389)
   CROSS_SECTION (CS70821,1.659)
   CROSS_SECTION (CS70561,1.919)
   CROSS_SECTION (CS70463,2.017)
   CROSS_SECTION (CS70418,2.062)
   CROSS_SECTION (CS70359,2.121)
   CROSS_SECTION (CS70269,2.211)
   CROSS_SECTION (CS70212,2.268)
   CROSS_SECTION (CS70141,2.339)
   CROSS_SECTION (CS70061,2.419)
   CROSS_SECTION (CS69926,2.554)
   CROSS_SECTION (CS69846,2.634)
   CROSS_SECTION (CS69646,2.834)
   CROSS_SECTION (CS69466,3.014)
   CROSS_SECTION (CS69255,3.225)
   CROSS_SECTION (CS69150,3.33)
   CROSS_SECTION (CS69049,3.431)
   CROSS_SECTION (CS68865,3.615)
   CROSS_SECTION (CS68744,3.736)
   CROSS_SECTION (CS68636,3.844)
   CROSS_SECTION (CS68509,3.971)
   CROSS_SECTION (CS68416,4.064)
   CROSS_SECTION (CS68317,4.163)
   CROSS_SECTION (CS68258,4.222)
   CROSS_SECTION (CS68026,4.454)
   CROSS_SECTION (CS67746,4.734)
   CROSS_SECTION (CS67460,5.02)
   CROSS_SECTION (CS67160,5.32)
   CROSS_SECTION (CS66839,5.641)
   CROSS_SECTION (CS66574,5.906)
   CROSS_SECTION (CS66242,6.238)
   CROSS_SECTION (CS66026,6.454)
   CROSS_SECTION (CS65697,6.783)
   CROSS_SECTION (CS65626,6.854)
   CROSS_SECTION (CS65433,7.047)
   CROSS_SECTION (CS65154,7.326)
   CROSS_SECTION (CS64743,7.737)
   CROSS_SECTION (CS64348,8.132)
   CROSS_SECTION (CS64153,8.327)
   CROSS_SECTION (CS63801,8.679)
   CROSS_SECTION (CS63510,8.97)
```

Fig. 12: Curved alignment of the cross sections.

The algorithm of the cross section interpolation is briefly sketched in the following. To grasp the meaning of the different parameters it is helpful to understand the basics of this interpolation algorithm.

This interpolation algorithm bases on the creation of spline curves (a spline is a special polynomial function which is often used for smooth interpolations between given points). For each fixpoint of the cross sections such a spline curve is determined which connects all the corresponding fixpoints with each other in a smooth way. Therefore every cross section must have the same number of fix points. Furthermore, the spline curves have the special property that they are aligned orthogonal to each cross section profile.
After the spline curves have been calculated, the positions of the new interpolated cross sections are determined in given intervals along the spline curves. As soon as these positions are known, the cross sections are created orthogonal to the tangent direction of the master spline. To determine the fixpoints of the new cross sections the intersections of this orthogonal cross section line with all spline curves (of the other fixpoints) are calculated. Finally the new cross section points are determined in between the fixpoints in a given transversal distance interval. The elevations of the cross section points are finally determined using a weighting procedure between the elevations of the left and the right cross section.


Cross Section Interpolation

Range
From Cross Section: CS63138 to CS62384

Longitudinal Spacing
Max distance between cross sections (m): 20

Transversal Spacing
Use max distance between points in interpolated cross section (m): [✓ Use local segment spacing]

Interpolation Alignment
Fixpoint Spline Alignment [✓ Interpolate Angle]

Spline Controls
Strength of spline orthogonality at fixpoints: 1.0
Master spline is based on: Fixpoint 1

Fig. 13: Setup dialog for the cross section interpolation

In *Fig. 13* the setup dialog for the interpolation is shown. The different parameters are explained briefly in the following. By clicking on *Interpolate* the interpolation of the cross sections finally starts if all data is available.

First of all the *Range* of the interpolation must be defined. This is done by specifying two subsequent cross sections which are chosen from a list of all existing cross sections in the drop down menus.

Another important parameter is the *Longitudinal spacing* which determines the resolution of the interpolated grid. Enter the maximum distance between two interpolated cross sections in [m]. If you choose a small value than many cross sections in small distances will be generated, if you choose a large value only few cross sections in large distances will be generated. The optimal choice depends on the type of simulation.

Furthermore, also the *Transversal spacing* can be specified. It determines the spacing of the points in the newly generated cross section profile. There are two possible choices here two determine the transversal spacing. You can either explicitly specify the maximum distance in [m] using the first option. Alternatively, by using the second option, the distance is chosen automatically from the left and right cross sections by the interpolation algorithm (*local segment spacing*).

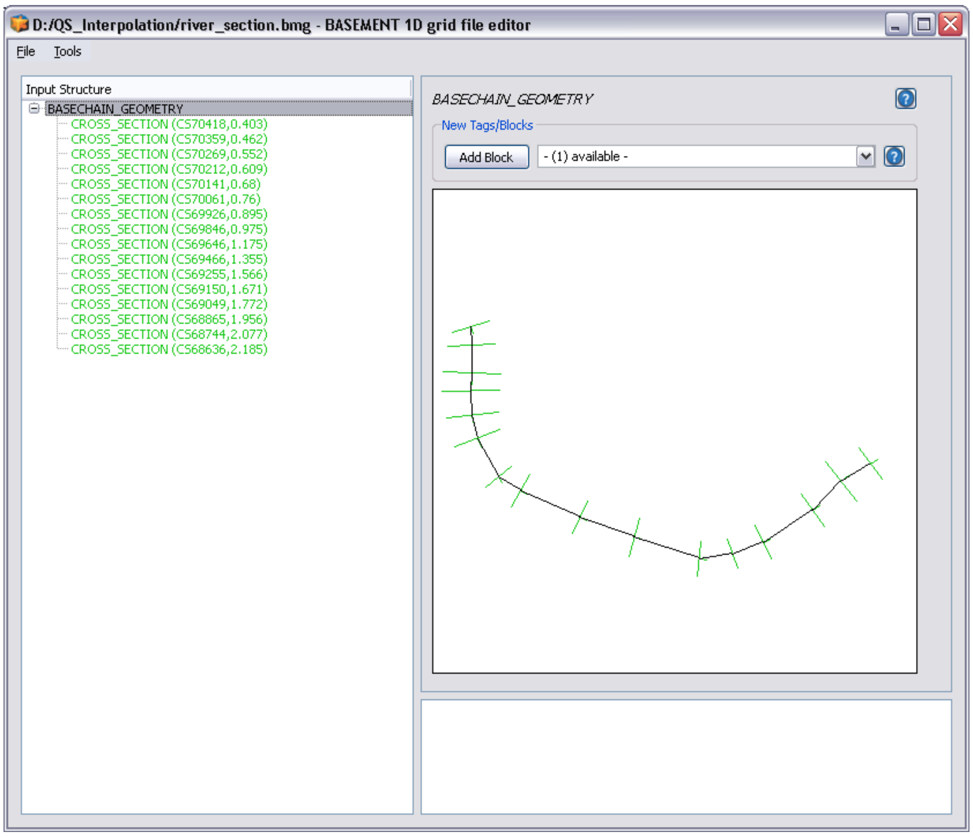
The alignment of the new cross sections in the x-y plane can be determined with two different methods in the *Interpolation Alignment* section. The *Fixpoint Spline Alignment* means that the cross sections are always oriented orthogonal to the master spine's tangent direction. On the other hand, using the option *Interpolate Angle* the orientation of the new cross section is determined by interpolation of the orientation angles of the left and the right cross sections. This latter option is recommended in strongly curved streams in order to prevent overlapping cross sections.

Finally in the *Spline controls* section some parameters of the spline calculations can be adapted to special needs. The *Strength of spline orthogonality* parameter determines if the spline always must be completely orthogonal to the cross sections or not. In strongly curved streams some relaxation from strict orthogonality (different from 1.0) may lead to nicer shaped spline curves. Some variations and iterative testing with this parameter may improve the interpolation result in such situations. And finally also the fixpoint which determines the master spline can be chosen. The master spline thereby is the spline which determines the orientation of the interpolated cross sections.

Please note: In order to generate a 2D mesh from a given 1D mesh, this interpolation option can be very helpful in combination with the *Export DTM* option.

4.3.6 Export DTM for BASEplane

This tool enables the user to convert a 1-D BASECHAIN_GEOMETRY (*Fig. 14*) into a digital terrain model (DTM) for further processing in SMS and BASEplane. The main application for this tool is to be found in combination with the Interpolation tool (see chapter 0): In a first step the cross sections are interpolated with the Interpolation tool in order to get a smooth river topography. In a next step the DTM is exported with the *Export DTM for BASEplane* tool (on the menu bar choose *Tools*). The generated DTM can be imported in SMS. Although the file is of .2dm type it can be easily converted into scatter points (DTM) in SMS. Then it can be used for the interpolation of the elevation information on any computational mesh.
Basically a computational mesh can be obtained directly from the *Export DTM for BASEplane* tool, if the interpolated cross sections are chosen in a close and optimal distance to each other. Nevertheless it is suggested to generate the mesh properly in SMS and to consider the generated DTM just as a terrain model from which to get the elevation information.



CROSS_SECTION (CS70418,0.403)
CROSS_SECTION (CS70359,0.462)
CROSS_SECTION (CS70269,0.552)
CROSS_SECTION (CS70212,0.609)
CROSS_SECTION (CS70141,0.68)
CROSS_SECTION (CS70061,0.76)
CROSS_SECTION (CS69926,0.895)
CROSS_SECTION (CS69846,0.975)
CROSS_SECTION (CS69646,1.175)
CROSS_SECTION (CS69466,1.355)
CROSS_SECTION (CS69255,1.566)
CROSS_SECTION (CS69150,1.671)
CROSS_SECTION (CS69049,1.772)
CROSS_SECTION (CS68865,1.956)
CROSS_SECTION (CS68744,2.077)
CROSS_SECTION (CS68636,2.185)

BASECHAIN_GEOMETRY

*Fig. 14: GUI of the BASEMENT 1D grid file editor. The 1-D BASECHAIN_GEOMETRY can be exported with **Export DTM for BASEplane** under the menu bar **Tools**.*

5 Built-In GUI Tools

In this section some built-in BUI tools are explained and information about the usage is provided. Built-in GUI Tools will pop up if a certain tag is activated by the user.

5.1 Interactive Visualization during run time using BASEviz

BASEviz is a small and lightweight visualization tool which can be used to visualize simulation results during run time. To activate the visualization tool, a *SPECIAL_OUTPUT* block of 'BASEviz' type must be created within the parent OUTPUT block. Then the BASEviz window will appear automatically by starting the simulation. The output can be visualized interactively using the mouse and keyboard keys according to the legend shown in the BASEviz window (see *Fig. 15* and *Fig. 16*). The view can be changed and the displayed variables can be selected. This visualization tool allows to easily check for a correct simulation setup and to stop a simulation run if some evident problems arise. Furthermore, it is possible to dump the rendered images from the visualization window in a JPEG image for a given time interval.

- BASEviz for 1-D simulations with BASEchain:

  All 1-D cross sections with its multiple slices are plotted one after the other along the x-axis. The water elevation is plotted within each cross section slice according to its present value.

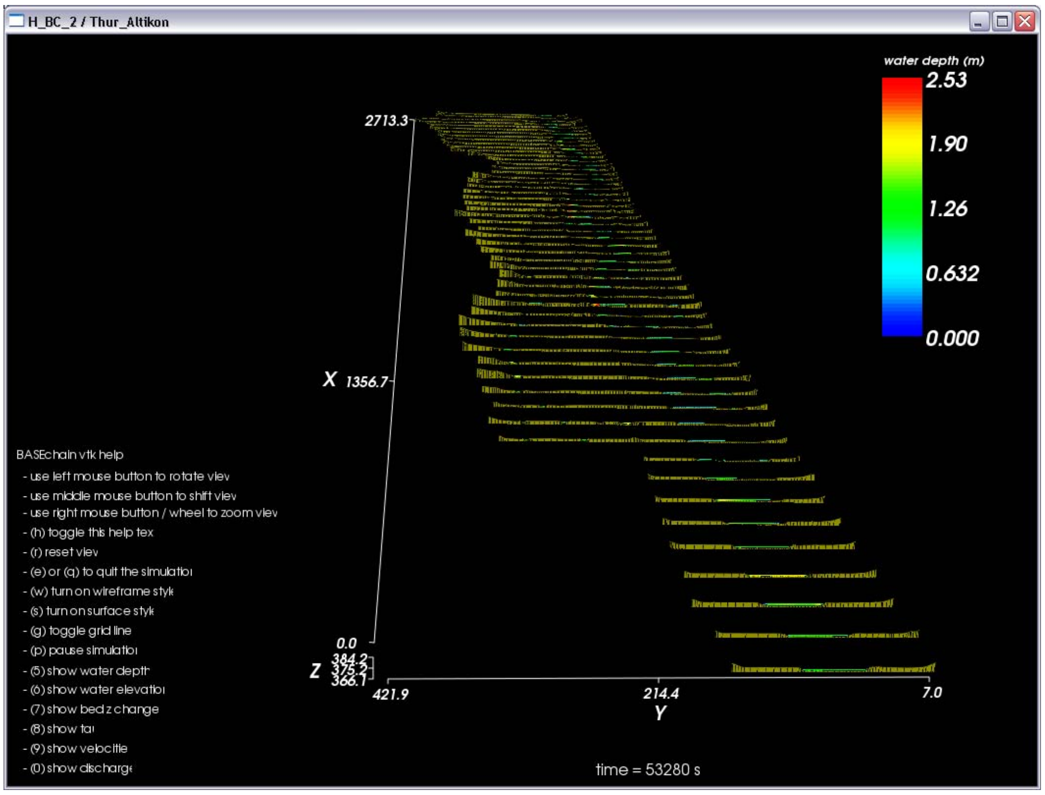
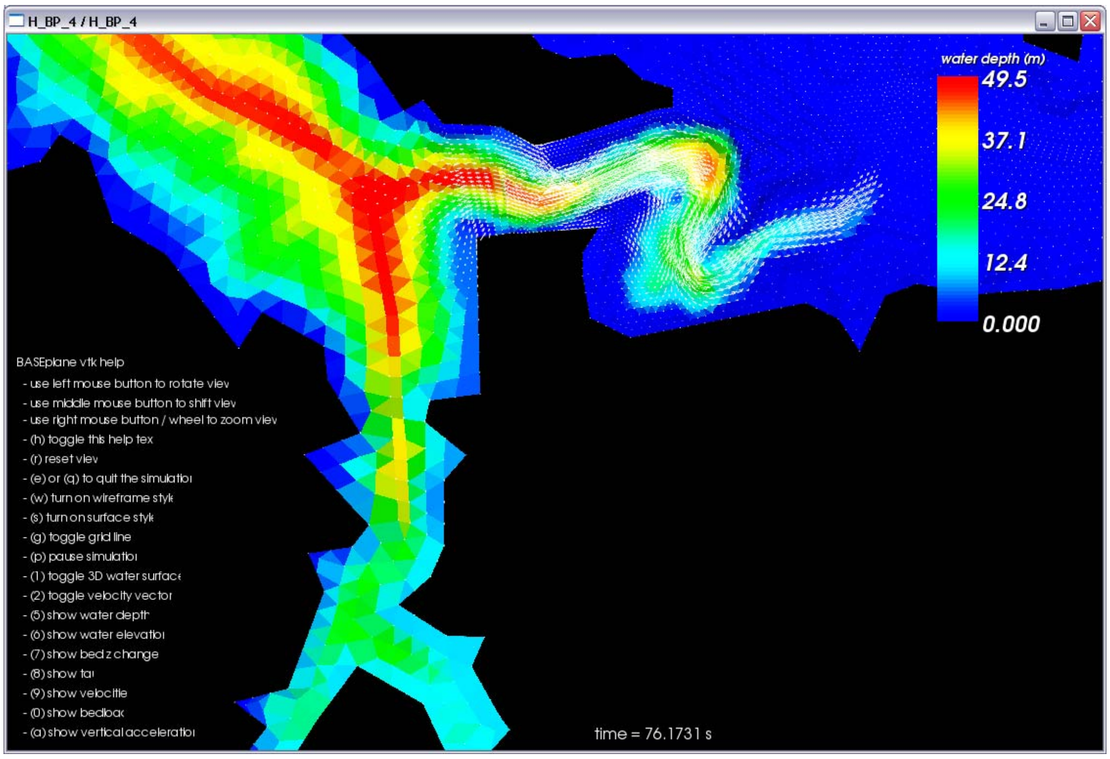


Fig. 15: Visualization of BASEchain with BASEviz

- BASEviz for 2-D simulations with BASEplane:

  The unstructured 2-D mesh is plotted in combination with a contour plot of a chosen output flow variable. Optionally, velocity vectors can be added to the data visualization.



BASEplane vtk help
- use left mouse button to rotate viev
- use middle mouse button to shift viev
- use right mouse button / wheel to zoom viev
- (h) toggle this help tex
- (r) reset viev
- (e) or (q) to quit the simulatio
- (w) turn on wireframe styl
- (s) turn on surface styl
- (g) toggle grid line
- (p) pause simulatio
- (1) toggle 3D water surfac
- (2) toggle velocity vector
- (5) show water depth
- (6) show water elevatio
- (7) show bed z change
- (8) show ta
- (9) show velocitie
- (0) show bedloa
- (a) show vertical acceleratio

Fig. 16: Visualization of BASEplane with BASEviz

BASEviz is based on the visualization libraries of the Visualization ToolKit (VTK, http://www.vtk.org) which makes use of OpenGL for rendering.

5.2 Manual Controller Interface (HID)

In order to create a controller a new CONTROLLER block is generated in the DOMAIN block. The HID controller provides an interface for the manual operation (*Fig. 17*). The control window will pop up automatically after starting the simulation with the start button. In the CONTROLLER block several manipulated variables and controlled variables can be defined. The manipulated variables will appear on the left hand side of the controller interface, whereas the monitored variables will show up on the right hand side (*Fig. 17*). In this example two manipulated variables (height of a weir and height of a gate) and two monitored variables (discharge over the weir and trough the gate) are selected. With the cursor the slide can be moved within the predefined range of the manipulated variable. Additionally the target value can be entered directly into the white text box (Target). As the simulation proceeds the impact on the monitored variable is visualized on the chart on the right hand side. Additionally the output and the impact of your control measures can be visualised with BASEviz (chapter 5.1).

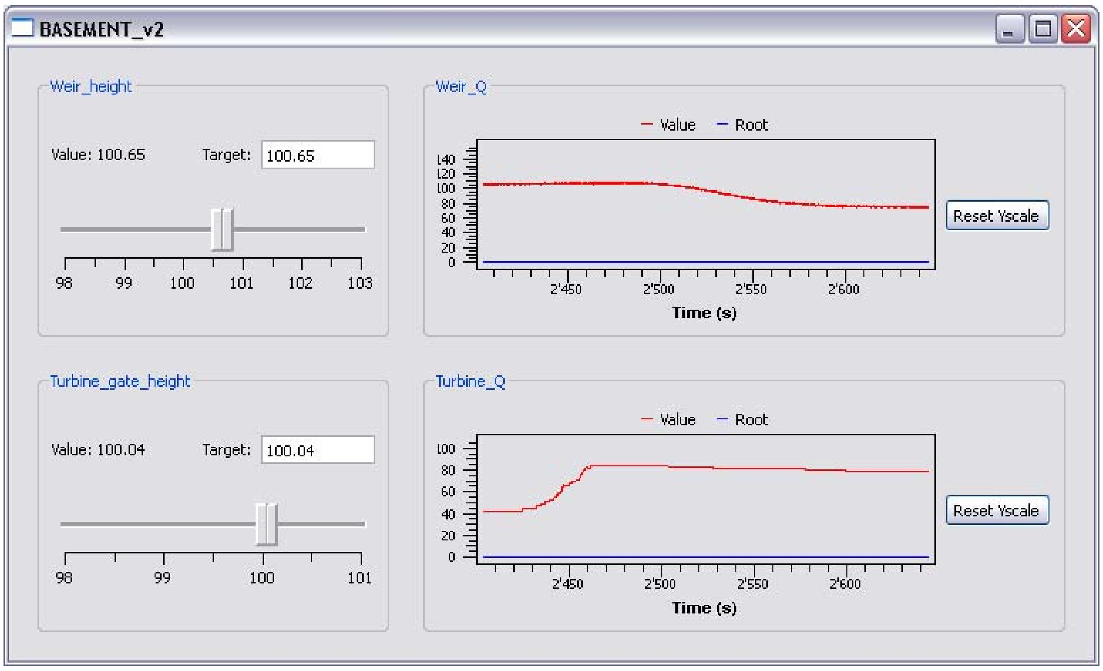


Fig. 17: Interface for the manual control and monitoring of the selected variables.

This page has been intentionally left blank.

uIV

TUTORIALS

of BASEMENT

This page has been intentionally left blank.

Table of Contents

1 General 1

2 Hydrodynamics and sediment transport at the river Thur

2.1 Introduction 2.1-1
2.1.1 General description 2.1-1
2.1.2 Used features 2.1-1
2.1.3 Purpose 2.1-2
2.2 Setting up the topography file 2.2-1
2.2.1 Cross sections 2.2-1
2.2.2 Definition of different cross section zones 2.2-3
2.2.3 Friction values 2.2-4
2.2.4 Computation of water surface elevation 2.2-5
2.2.5 Characterisation of the sediments 2.2-6
2.2.6 Define flowing zones 2.2-7
2.3 Setting up the command file 2.3-1
2.3.1 Project 2.3-1
2.3.2 Domain 2.3-1
2.3.3 Define the physical properties 2.3-1
2.3.4 One dimensional simulation 2.3-1
2.3.4.1 Define the geometry 2.3-2
2.3.4.2 Define hydraulic information 2.3-2
2.3.4.2.1 Define the upper boundary condition 2.3-2
2.3.4.3 Define the lower boundary condition 2.3-3
2.3.4.3.1 Define initial condition: 2.3-3
2.3.4.3.2 Define default friction values 2.3-3
2.3.4.3.3 Declare parameters for hydraulic computation 2.3-4
2.3.4.4 Define output 2.3-4
2.4 Perform hydraulic simulations 2.4-1
2.4.1 Perform steady flow simulation (Thur1) 2.4-1
2.4.2 Perform simulation of the floods (Thur2) 2.4-3
2.5 Complete the command file for bed load transport 2.5-1
2.5.1 Define the bed material 2.5-1
2.5.2 Soil assignment 2.5-2
2.5.3 Define general parameters for sediment transport 2.5-3
2.5.4 Define specific parameters for bed load transport 2.5-4
2.5.5 Define boundary conditions for bed load 2.5-4
2.5.6 Generate a “geometry” file 2.5-5
2.6 Perform bed load simulation (Thur 3) 2.6-1
2.7 Hydraulic computation using HEC-RAS 3.1.3 .... 2.7-1

3 Hydrodynamics and sediment transport at the river Flaz

3.1 Introduction 3.1-1
3.1.1 Case study description 3.1-1
3.1.2 Tutorial structure 3.1-2
3.2 Computational grid 3.2-1
3.3 Setting up the command file 3.3-1
3.3.1 Project 3.3-1
3.3.2 Domain 3.3-2
3.3.3 Parallel 3.3-2
3.3.4 Physical properties 3.3-2
3.3.5 Two dimensional simulation 3.3-3
3.3.5.1 Geometry 3.3-3
3.3.5.1.1 Define the hydraulics 3.3-5
3.3.5.1.2 Hydraulic boundary conditions 3.3-5
3.3.5.1.3 Initial condition 3.3-6
3.3.5.1.4 Friction 3.3-6
3.3.5.1.5 Computational parameters 3.3-7
3.3.5.2 Define the output 3.3-7
3.4 Perform hydraulic simulation 3.4-1
3.4.1 Perform steady flow simulation 3.4-1
3.4.2 Perform unsteady flow simulation 3.4-2
3.4.3 Calibration of the hydraulic model 3.4-6
3.5 Morphological simulation with single-grain bed load transport 3.5-1
3.5.1 Define the morphological information 3.5-1
3.5.1.1 Morphological parameters 3.5-2
3.5.1.2 Bed material 3.5-2
3.5.1.3 Grain size distribution 3.5-2
3.5.1.4 Grain mixture 3.5-3
3.5.1.5 Define the soil composition 3.5-3
3.5.1.6 Fixed bed elevation 3.5-4
3.5.1.7 Assignment of the defined soil types 3.5-5
3.5.1.8 Initial condition 3.5-5
3.5.1.9 Bed load boundary condition 3.5-5
3.5.1.10 Bed load parameter 3.5-6
3.5.1.11 Gravitational transport 3.5-6
3.5.2 Define the output 3.5-8
3.6 Perform morphological simulation with single-grain bed load transport ... 3.6-1
3.7 Morphological simulation with multi-grain bed load transport 3.7-3
3.7.1.1 Morphological parameters 3.7-3
3.7.1.2 Grain size distribution 3.7-3
3.7.1.3 Grain mixture 3.7-4
3.7.1.4 Define the soil composition 3.7-4
3.7.1.5 Bed load boundary condition 3.7-6
3.7.1.6 Bed load parameter 3.7-6
3.7.1.7 Define the output 3.7-7

3.8 Perform morphological simulation with multi-grain bed load transport 3.8-9

List of Figures

Fig. 1: View of the simulated river section 2.1-1
Fig. 2: Cross section points inserted in the topography editor....................2.2-2
Fig. 3: Delimitation of cross section zones....................2.2-3
Fig. 4: Declaration of soil types 2.2-6
Fig. 5: Definition of non flowing areas 2.2-7
Fig. 6: Longitudinal profile 2.4-2
Fig. 7 Hydrograph of the flood event of 2005 ....2.4-3
Fig. 8: Longitudinal profile for maximum discharge 2.4-5
Fig. 9: Example of resulting water surface elevation....................2.4-6
Fig. 10: Soil assignment in the user interface 2.5-2
Fig. 11: Longitudinal profile of mean bottom level....................2.6-1
Fig. 12: Cross section....................2.6-1
Fig. 13: Different morphological river subsections of the new section of the river Flaz. 3.1-1
Fig. 14: Computational mesh generated by SMS for a section in the widening part. The bold black lines are break lines for the mesh. 3.2-1
Fig. 15: Material indexes (Ids) used for assignment the friction factor. 3.2-2
Fig. 16: Node id numbers for the definition in the STRING_DEF block for the inflow boundary.3.3-4
Fig. 17: Node id numbers for the definition in the STRING_DEF block for the outflow boundary.3.3-4
Fig. 18: Stationary hydrograph file saved as Inflow_stationary.txt. 3.3-6
Fig. 19: Steady inflow hydrograph and outflow hydrograph. 3.4-1
Fig. 20: Flow depth and flow velocity vectors at the steady state of the model....................3.4-2
Fig. 21: Hydrograph of the flood event of July 2004....................3.4-3
Fig. 22: Inflow hydrograph stored in the file Inflow_instationary.txt. Note that the points (...) are just illustrative in order to show the first and last line of the file. 3.4-4
Fig. 23: Maximal flow depth and flow velocity vectors of the unsteady flow simulation. 3.4-6
Fig. 24: Modeled bed elevation (z_bed) and two cross sections defined in the widening part......3.6-1

Fig. 25: Comparison of the river bed before and after the flood event in cross-section QS 1 and QS 2. ....3.6-2

Fig. 26: Changes of the morphology (deltaz) due to the flood event with the single-grain model. The red colour range represents deposition and the blue colour range shows erosion..........3.6-2

Fig. 27: Changes of the morphology (deltaz) due to the flood event with the multi-grain model. The red colour range represents deposition and the blue colour range shows erosion..........3.8-9

IV TUTORIALS

1 General

The tutorials of BASEMENT provide a step-by-step introduction to numerical modelling. Therefore, all necessary parts of the simulation procedure, such as setting up the topography or defining the simulation’s command file, are described in detail herein.

This part of the system manuals lives with your feedback! Please don’t hesitate to submit further examples, which you think could be of interest for other users. Also feel free to let us know, what you miss and would like to study in the next version of the tutorials. Please use the official BASEMENT email, basement@ethz.ch, therefore. Many thanks for your collaboration!

This page has been intentionally left blank.

IV TUTORIALS

Part 1: Applications of BASEchain

2 Hydrodynamics and sediment transport at the river Thur

2.1 Introduction

2.1.1 General description

This Tutorial describes the necessary steps for the simulation of hydrodynamics and bed load in a specific section of the river Thur. In the considered section a river widening has been realized during the last years. It’s located in “Altikon” and illustrated in the photo below. The flow direction is from right to left. The bed modification over a year including an important flood will be simulated.

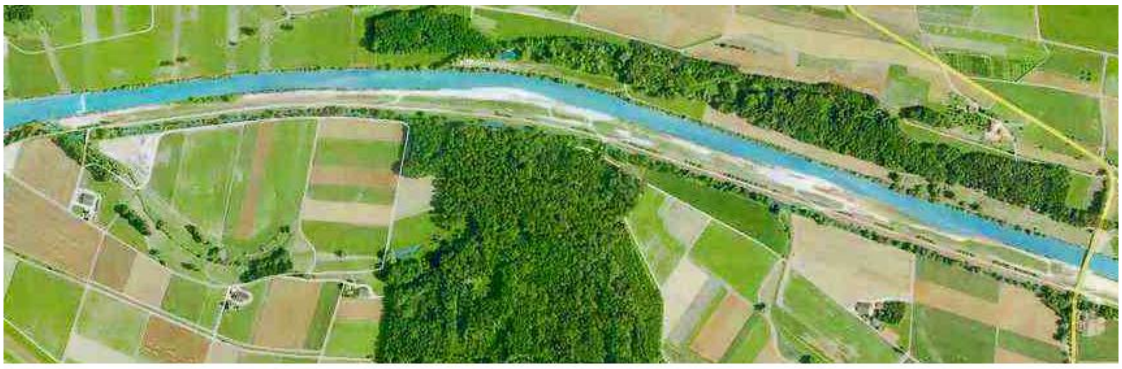


Fig. 1: View of the simulated river section

2.1.2 Used features

In this tutorial will be treated the following points:

- Preparation of the needed input files;
- Simulation of a steady flow to use for the following simulations;
- Use of composite cross sections;
- Simulation of bed load with formula of Meyer-Peter Müller;
- Use of dry initial condition;

- Use of a file to define the initial conditions;
- Use of the following boundary conditions:
    - Inflow hydrograph
    - Inflow of sediment in/out
    - Outflow h-q relation
    - Outflow of sediment in/out
- Representation of the results.

2.1.3 Purpose

In the year 2005, intensive rainfall led to a large flood event. The aim of the simulations in this tutorial is to study, which influence this flood had on the geometry of the channel of the river Thur.

2.2 Setting up the topography file

2.2.1 Cross sections

The data of the topography are available in the form of cross section measurements, where each measured point is given by its x, y and z coordinates.
This is an extract from the raw data:

```
x               y               z
698578.504      272450.223      376.841
698578.494      272446.999      374.991
698578.32       272444.286      373.748
698577.929      272441.889      373.229
698578.081      272439.244      372.207
698578.533      272437.229      371.544
698578.56       272434.366      370.869
698578.612      272431.522      370.766
698578.86       272429.064      370.401
698579.201      272426.526      370.388
698579.323      272424.56       370.617
698578.937      272422.91       370.341
698579.208      272421.645      370.436
…
```

These data have to be separated in groups belonging to one cross section and then transformed in a way to have a z(y) relation, where the smallest y is the extreme point on the left river side.

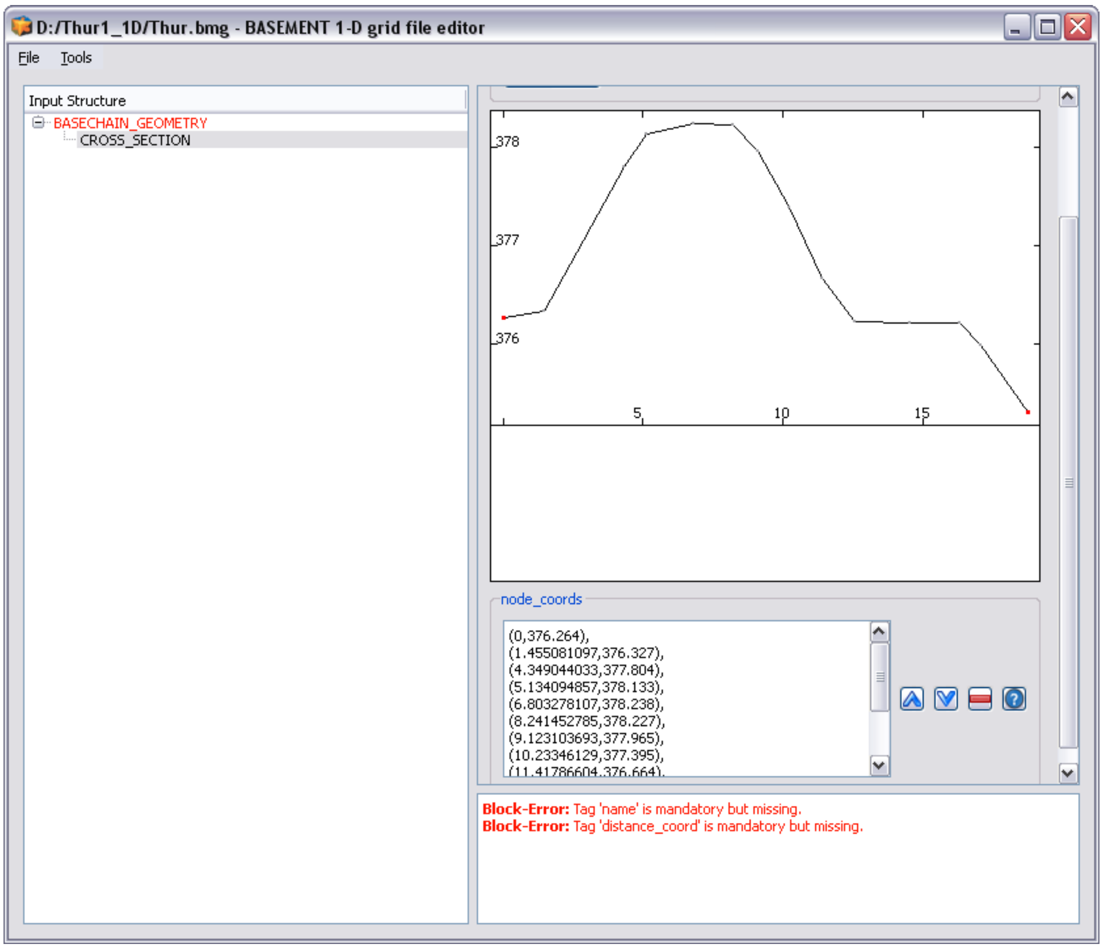
Example:

```
y               z
0               376.264
1.455081097     376.327
4.349044033     377.804
5.134094857     378.133
6.803278107     378.238
8.241452785     378.227
9.123103693     377.965
10.23346129     377.395
11.41786604     376.664
12.57016281     376.221
14.53240603     376.21
16.34176138     376.215
17.09961488     375.99
18.74552432     375.296
...
```

Additionally the distance from the upstream end of the channel (first cross section) has to be determined for each cross section.

The obtained geometry points can be introduced in the topography by copy pasting it directly into the node coordinates field in the grid file editor. The editor will translate the column wise data into the proper syntax. Another, more efficient way is to use the python scripts available on www.basement.ethz.ch to transform topography data in excel format into the BASEMENT format in a first step.

Example:



```
(0,376.264),
(1.455081097,376.327),
(4.349044033,377.804),
(5.134094857,378.133),
(6.803278107,378.238),
(8.241452785,378.227),
(9.123103693,377.965),
(10.23346129,377.395),
(11.41786604,376.664),
```

Block-Error: Tag 'name' is mandatory but missing.
Block-Error: Tag 'distance_coord' is mandatory but missing.

Fig. 2: Cross section points inserted in the topography editor

The minimum information we have to provide for each cross section besides the node coordinates are the cross section name and the distance coordinate measured from upstream to downstream in km.

2.2.2 Definition of different cross section zones

To reduce some drawbacks of the 1-D simulation, in the present case it is useful to define a main channel and flood plains, as well as the bed bottom to which is limited the bed load transport.

In the figure below, the flood plains are given by the part of the cross section not defined as main channel. The soils by their indexes. The keys 2 or 1 refer to the type of soil which is defined later in the command file. Here we use only one soil for the whole bottom, but it is also possible to add several soils of the same type or of different types as shown in figure *Fig. 3.* Further, different friction values can be defined for different parts of the cross section. The active range should span from the left to the right dike. Points outside the active range are simply ignored.

The graphical view of the cross section data helps to identify the correct point and set the ranges to the correct lateral node coordinates. For convenience, one can switch into the text editor mode of the input file by choosing from the *Tools* Menu the option *Edit Raw.*

Example:

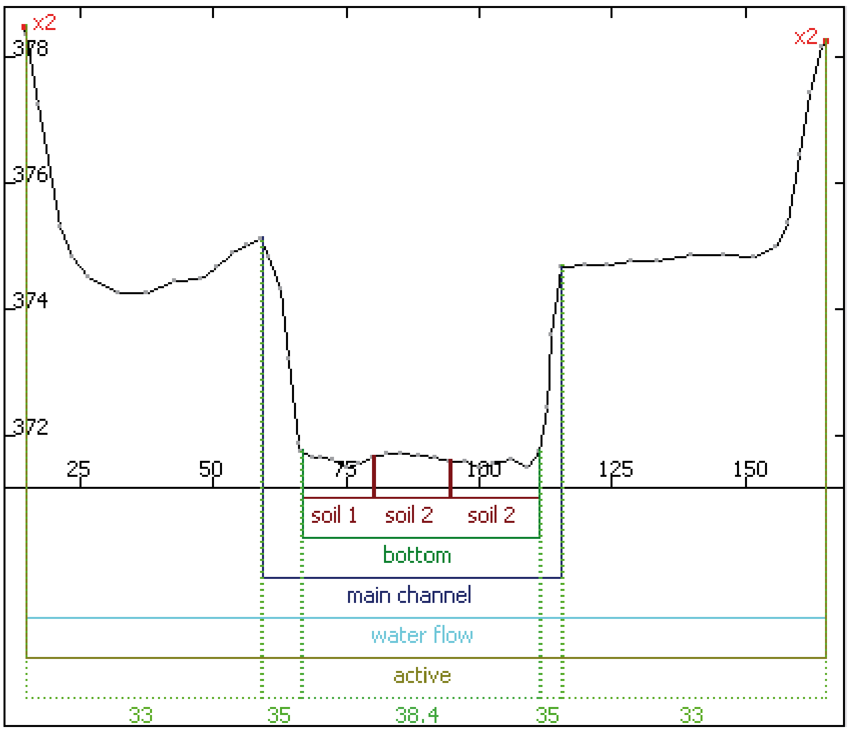
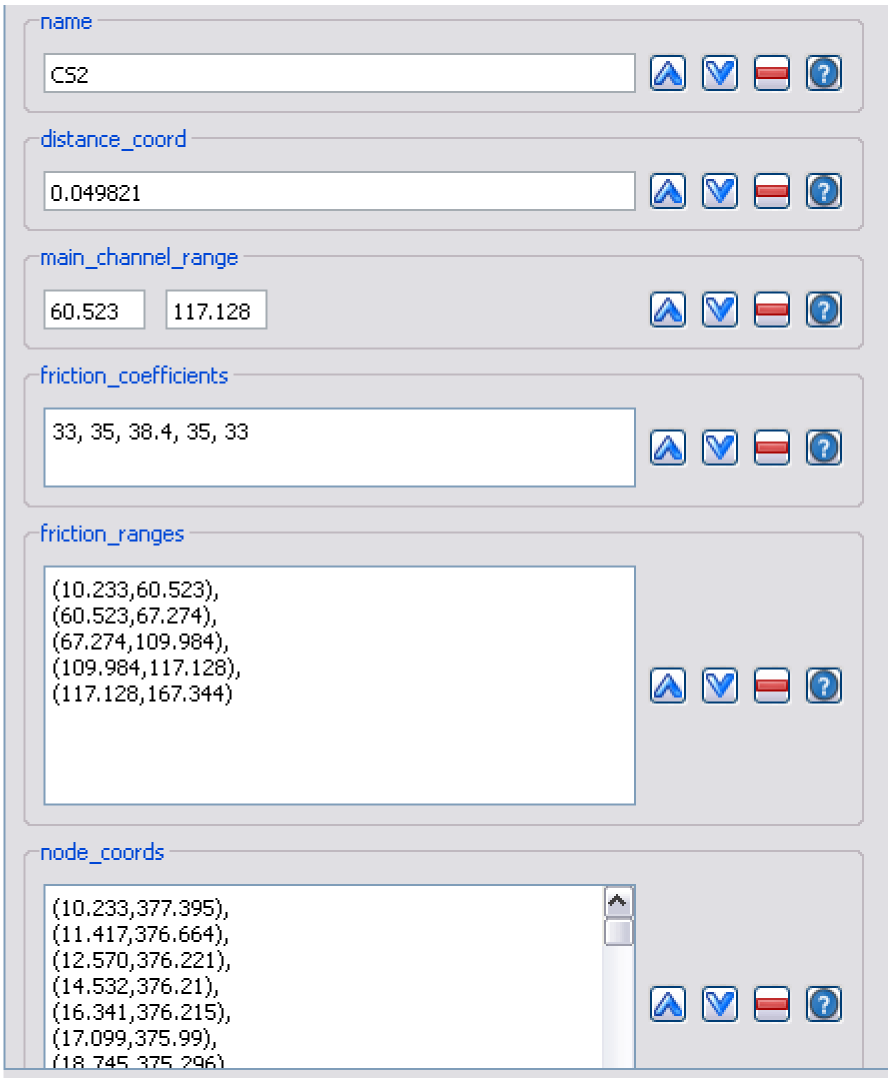


Fig. 3: Delimitation of cross section zones.

Example:



CS2

0.049821

60.523 117.128

33, 35, 38.4, 35, 33

(10.233,60.523),
(60.523,67.274),
(67.274,109.984),
(109.984,117.128),
(117.128,167.344)

(10.233,377.395),
(11.417,376.664),
(12.570,376.221),
(14.532,376.21),
(16.341,376.215),
(17.099,375.99),

Fig. 4: Definition of cross section properties

2.2.3 Friction values

For the friction determination the Strickler approach is used. This is declared in the command file by setting the *type* in the FRICTION block within the HYDRAULICS section to *strickler*. In this case, Strickler k-values have to be defined for the different regions. The banks of the main channel are partially covered with small bushes. The flood plains are covered with grass, stones and sand, but there are also zones with trees.

The following k_{Str} values are used:

- Banks of main channel: $k_{Str} = 35$ [$m^{1/3}/s$]
- Flood plains: $k_{Str} = 33$ [$m^{1/3}/s$].

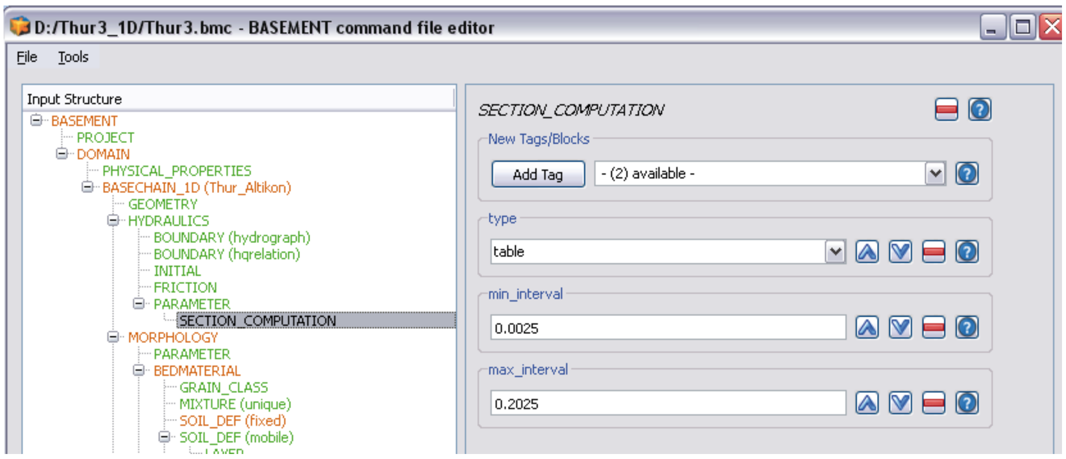
For the bed bottom the following transformation, based on the grain characteristics of the sediment is used (d90 = 5 cm):

Bottom: $k_{St} = \frac{23}{\sqrt[6]{d_{90}}} = 38$ [$m^{1/3}/s$].

In BASEMENT internally, the cross section is represented by slices, defined by the segment between two nodes. Each slice has its own properties. Therefore we have to provide so called ranges to assign the friction values to the respective slices. The ranges can be defined either referring to node coordinates (note that you have to match the coordinates exactly) or by referring to slice indexes, starting at index 1 from left to right.

2.2.4 Computation of water surface elevation

As it is much quicker, the use of tables for the computation of the water surface elevation and other hydraulic variables is chosen for this example. In the case of tables, all properties are pre-computed for a given set of points and only updated in case of a non-negligible change of the soil. This is accomplished using the block SECTION_COMPUTATION in PARAMETER. As all variables are calculated for several water surface elevations, the maximum and minimum intervals between the different levels have to be set accordingly. The default spacing is given by max_interval – min_interval. Whenever the bed changes, the table is updated accordingly.


D:/Thur3_1D/Thur3.bmc - BASEMENT command file editor

File Tools

Input Structure
- BASEMENT
  - PROJECT
  - DOMAIN
    - PHYSICAL_PROPERTIES
    - BASECHAIN_1D (Thur_Altikon)
      - GEOMETRY
      - HYDRAULICS
        - BOUNDARY (hydrograph)
        - BOUNDARY (hqrelation)
        - INITIAL
        - FRICTION
        - PARAMETER
          - SECTION_COMPUTATION
      - MORPHOLOGY
        - PARAMETER
        - BEDMATERIAL
          - GRAIN_CLASS
          - MIXTURE (unique)
          - SOIL_DEF (fixed)
          - SOIL_DEF (mobile)

SECTION_COMPUTATION

New Tags/Blocks: Add Tag - (2) available -

type: table

min_interval: 0.0025

max_interval: 0.2025

Fig. 4: Definition of table values

2.2.5 Characterisation of the sediments

Two types of ground will be defined:

1. the ground is not erodible;
2. the ground is composed by sediments with a mean diameter of 2.5 cm.

In the topography file the codes of the different types are assigned to different cross sections. The code can be set in the cross sections by creating a new sub block SOIL_DEF, where the index is set to the respective soil index in the command file and the span of the soil is defined via the range it extends.

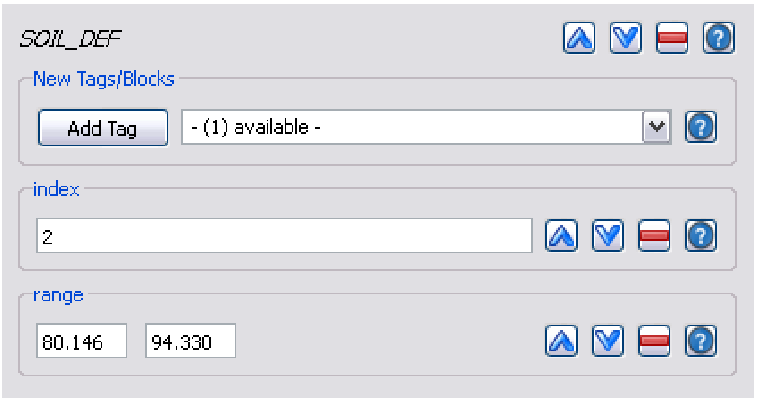

SOIL_DEF
index: 2
range: 80.146 94.330

Fig. 4: Declaration of soil types

2.2.6 Define flowing zones

The 1-D model considers the flow velocity to be the same over the whole width of the cross section: This is obviously not true, especially for cross sections where important zones are behind a sort of dike, like it occurs very often at the Thur. This effect has an important influence on the bed load transport. For this reason, regions where the water does not flow are declared using the water_flow_range tag. The next figure shows an example of a cross section with the different zones. Of course, we only mention this here. The tutorial topology already contains the required ranges.

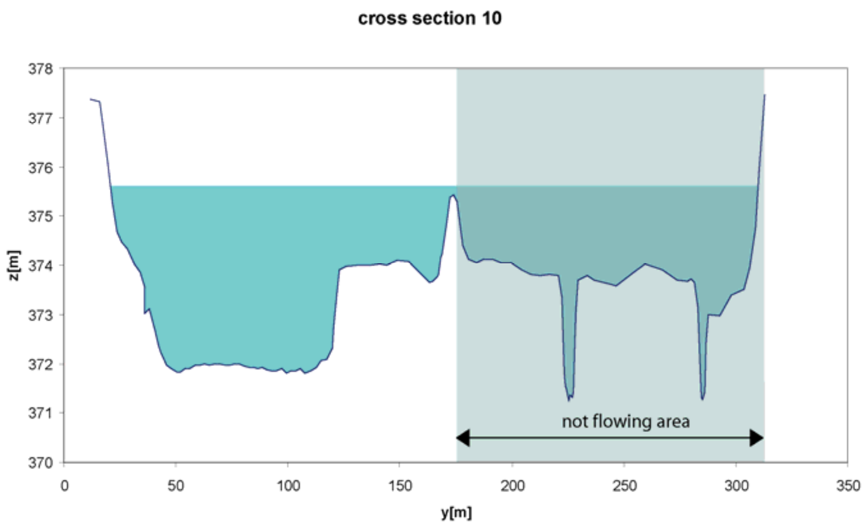


Fig. 5: Definition of non flowing areas

This page has been intentionally left blank.

2.3 Setting up the command file

2.3.1 Project

The first command file is called Thur1.bmc.
The first step is to define a project by its name, the author and the date:

```
PROJECT
{
     title = Thur
     author = rm
     date = 10.8.2006
}
```

2.3.2 Domain

Then a domain is defined in which all parameters concerning this computation are defined. The first parameter is the name of the computation region.

```
DOMAIN
{
     multiregion = Thur
     […]
}
```

2.3.3 Define the physical properties

The Physical properties normally do not change from one project to another.

```
PHYSICAL_PROPERTIES
{
     gravity = 9.81
     viscosity = 0.000001004
     rho_fluid = 1000
}
```

2.3.4 One dimensional simulation

The next step is to declare a *BASECHAIN_1D* block. This will make the program execute a 1-D simulation. The name of the computational region is given here.

```
BASECHAIN_1D
{
     region_name = Thur_Altikon
}
```

2.3.4.1 Define the geometry

The next block defines in which file the topography is stored and which type of geometry file is used. The cross section names are listed from upstream to downstream.

```
GEOMETRY
{
      type = basement
      file = ThurTopo.bmg
      cross_section_order = ( CS1 CS2 CS3 …….CS54 CS55 )
}
```

2.3.4.2 Define hydraulic information

All information concerning the hydraulic simulation is declared in the block *HYDRAULICS*.

```
HYDRAULICS
{
      […]
}
```

2.3.4.2.1 Define the upper boundary condition

The upper boundary condition is defined by a hydrograph, which is stored in a separate file. Indicate also the precision with which the discharge corresponding to the iteratively determined area must correspond to the given discharge and the maximum number of iterations allowed to reach this precision. The slope of the first cross section must be given in per-mille (the last 3 values are used only in case of supercritical flow).

```
BOUNDARY
{
      type = hydrograph
      string = upstream
      file = ThurSteadyHydrograph.txt
      precision = 0.001
      number_of_iterations =100
      slope = 0.93
}
```

2.3.4.3 Define the lower boundary condition

The lower boundary is an h-q- relation which is calculated internally. Again, we have to define the boundary condition with a specific slope.

```
BOUNDARY
{
      type = hqrelation
      string = downstream
      slope = 1.5
}
```

2.3.4.3.1 Define initial condition:

The channel is considered to be initially dry. Note that starting with a dry channel is for the depth-average equations a numerically delicate problem. So this option will require some care when we set the numerical parameters.

```
INITIAL
{
      type = dry
}
```

2.3.4.3.2 Define default friction values

The declaration of a default friction type and a default friction value are mandatory. The friction values are overwritten by the values declared in the topography file.

```
FRICTION
{
      type = strickler
      default_friction = 35
}
```

2.3.4.3.3 Declare parameters for hydraulic computation

For the first computation the simulation time is set to 15000 s. For a computation on a dry bed a small initial time step should be chosen. It is used only at the very beginning, as there is no flow in the channel from which the time step could be deduced. The maximum time step should be bigger as all time steps computed during the simulation.

```
PARAMETER
{
      total_run_time = 15000
      initial_time_step =  1
      maximum_time_step = 60
      CFL = 0.95
      minimum_water_depth = 0.0001
}
```

Define the hydrograph file named ThurSteadyHydrograph.txt:

```
// T        Q
0          30
100000     30
```

2.3.4.4 Define output

If only standard output is needed, only the time step for file printing and for console printing has to be defined. If Tecplot software is available, it is also very useful to generate a tecplot file.

```
OUTPUT
{
      output_time_step = 100
      console_time_step = 100
      SPECIAL_OUTPUT
      {
            type = tecplot_all
            output_time_step = 100
      }
}
```

2.4 Perform hydraulic simulations

2.4.1 Perform steady flow simulation (Thur1)

The first simulation has the aim to create a steady flow as initial condition.
Place the 3 files in the same folder and start the simulation by double-clicking on the command file Thur1.bmc or by loading it as command file in the graphical user interface.

When the simulation has terminated, open the file named “restart” using a text-editor such as notepad. If the discharges Q correspond to the steady inflow discharge in all cross sections, then the steady state has been reached. The restart file can now be used as an initial condition file for the next step. Set the time to 0. For this purpose, it must be saved as a new text file, because the filename it currently has is used in subsequent iterations for the new restart file. Therefore, save the restart file using the name InitialThur.txt

```
CS41      25.785905     30
CS42      24.429422     30
CS43      22.647555     30
CS44      26.453984     30
CS45      34.367756     30
CS46      35.536409     30
CS47      39.03467      30
CS48      38.658572     30
CS49      38.016649     30
CS50      37.518627     30
CS51      34.37052      30
CS52      47.002793     30
CS53      41.384505     30
CS54      39.940648     30
CS55      31.36507      30
```

You should also have a look at the file named ThurTopoout.txt and verify whether there have been any errors or strange values resulting from your topography file. (Of course, the grid file you have here in the tutorial should not have any errors. It’s just a good exercise to check your geometry if the program does not work as expected).
You should also have a look at the main output file to see if there appear suspicious values, which could indicate that there is an error somewhere in your model setup.

Now, take from the output file the columns named distance, zbed, z and eline for the last time step and use them to plot a longitudinal profile (e.g. with Excel).

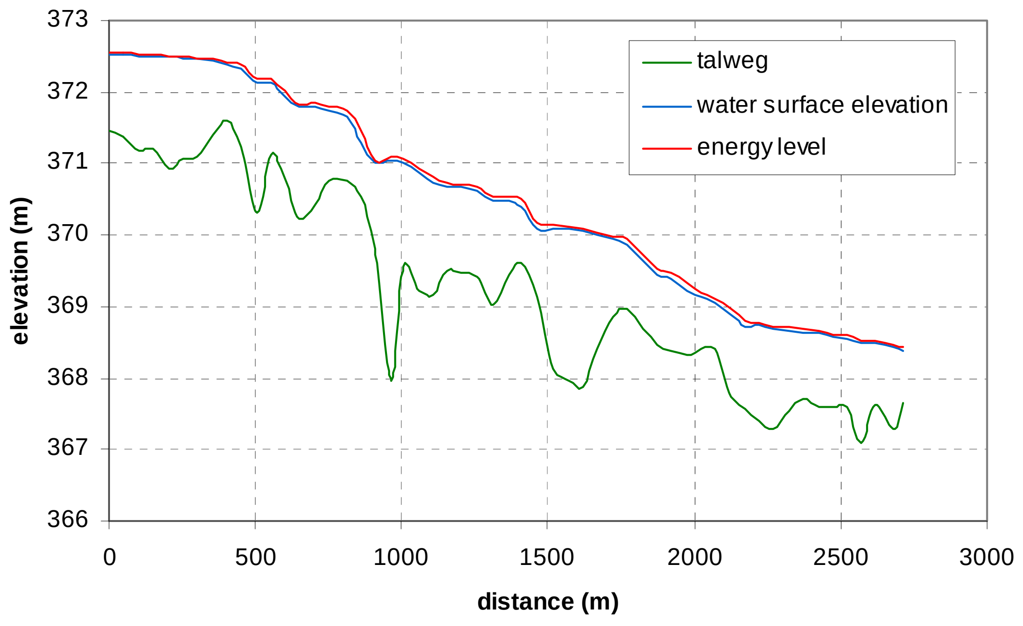


Fig. 6: Longitudinal profile

As it seems that there is no problem, we can proceed to the next step.

2.4.2 Perform simulation of the floods (Thur2)

Copy the following files in a new Folder (or take the ones in the second zip file):

```
Thur1.bmc
ThurTopo.txt
InitialThur.txt
```

Extract the ThurHydrograph.txt from the second tutorial zip file. It contains the hydrograph of the flood event in 2005.

```
T         Q
0         30.648
3600      34.05
7200      37.305
10800     39.707
14400     41.18
18000     41.916
21600     42.275
25200     42.654
28800     43.646
32400     45.444
...
```

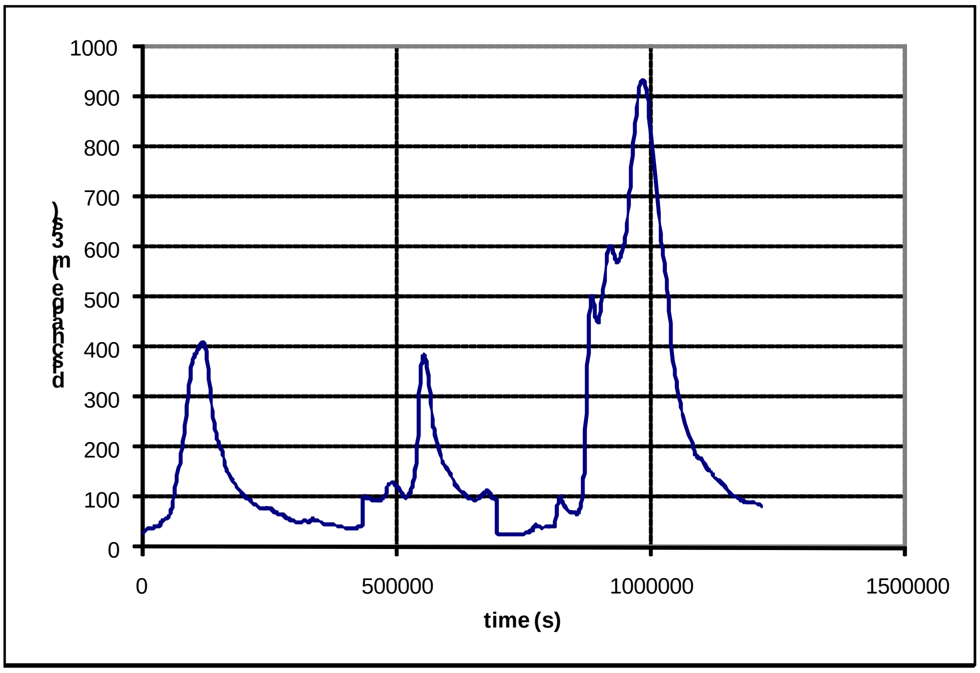


Fig. 7 Hydrograph of the flood event of 2005

Rename the command file Thur2.bmc. For the upstream boundary condition, change the data file into *ThurHydrograph.txt*.

```
BOUNDARY
{
     boundary_type = hydrograph
     boundary_area = upstream
     boundary_file = ThurHydrograph.txt
     precision = 0.001
     number_of_iterations =100
}
```

For the initial condition, we start now from the restart file of the last calculation:

```
INITIAL
{
     initial_type = fileinput
     initial_file = InitialThur.txt
}
```

The simulation now lasts longer, 338 hours. So we change it in the parameter section. Beside this we increase the initial_time_step to 3.

```
PARAMETER
{
     total_run_time = 1216800
     initial_time_step = 3
     CFL = 0.95
     minimum_water_depth = 0.01
     maximum_time_step = 60
}
```

The output will be plotted less often. Therefore, change the console and output time.

```
OUTPUT
{
     output_time_step = 1000
     console_time_step = 1000
}
```

Run the file Thur2.bmc. On my computer, this takes about 23 seconds (which means we calculate 50’000 real time seconds in 1).

When the simulation is finished, have a look at the Thur2out file, take the columns of distance, zbed, z and eline for the time of maximal discharge (933 m^3/s): 982000 s (= 273 hours) and plot them.

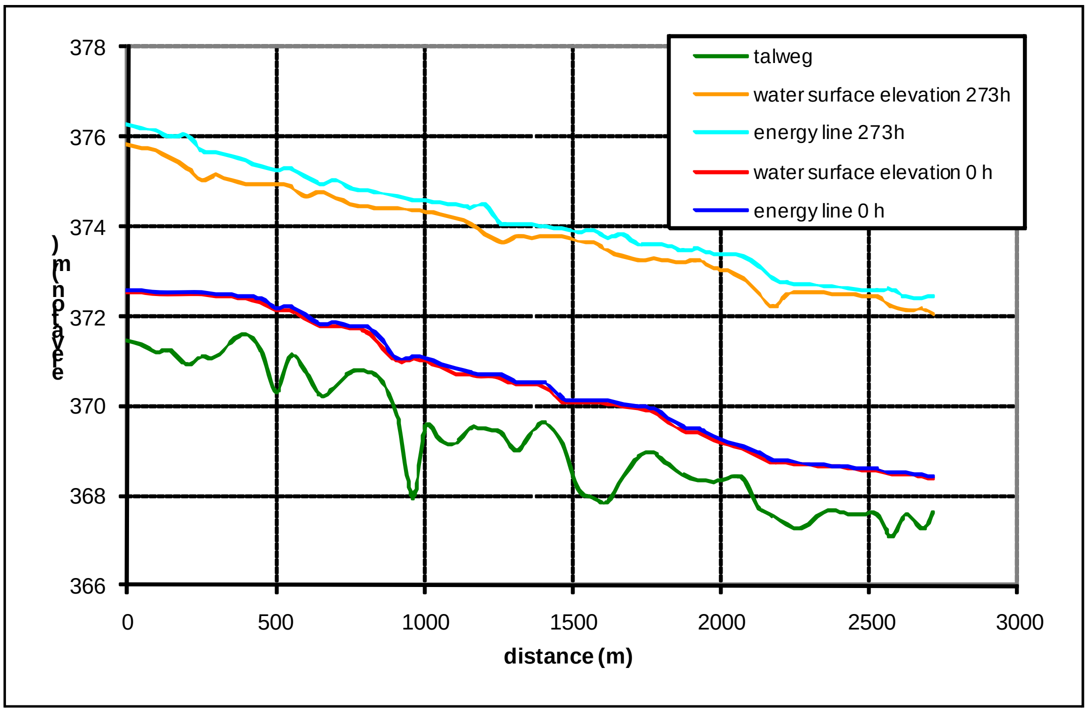


Fig. 8: Longitudinal profile for maximum discharge

Then plot some interesting cross sections with their water surface elevation, for the same time. This can help to see what happens, and which parts of the cross sections are touched by the flood. For this purpose the cross section geometry and can be taken from the topography file and the water surface elevation from the main output file. Alternatively a monitoring point of type “geometry” could be used.

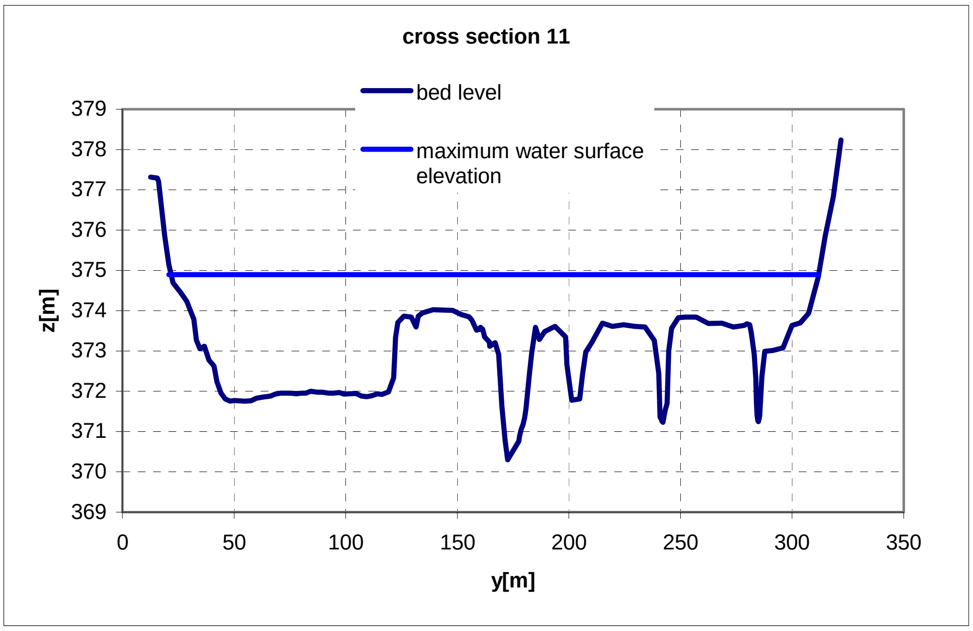


Fig. 9: Example of resulting water surface elevation

2.5 Complete the command file for bed load transport

After copying the files from simulation 2 in a new folder the command file must be completed with information about bed load, which is grouped in the block MORPHOLOGY. The new command file is renamed Thur3.bmc.
In the block BASECHAIN-1D chose MORPHOLOGY and press Add Block. Go in the block MORPHOLOGY.

2.5.1 Define the bed material

The simulation is executed with a single grain class with mean diameter = 2.5 cm. This means that you have to define one grain class and one mixture. Add a block of type GRAIN_CLASS and one of type MIXTURE. In the GRAIN_CLASS block add the diameter.
In the MIXTURE block add the name and the volume fraction.

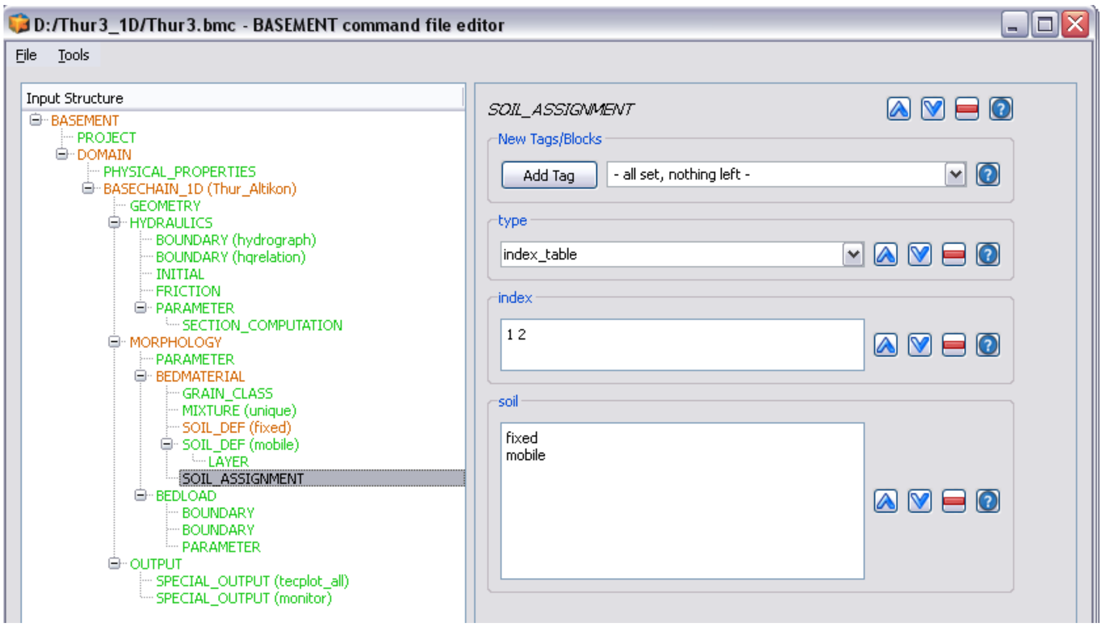
```
BEDMATERIAL
{
      GRAIN_CLASS
      {
            diameters = ( 25 )
      }
      MIXTURE
      {
            name = unique
            volume_fraction = ( 100 )
      }
      [...]
}
```

Two types of soils are defined: one which is fixed (code 1)and one with a sub layer of 5 m thickness which is attributed to the bed bottom where bed load takes place (code 2). Add twice a block of type SOIL_DEF. The first one needs only a name as it has no layers of material. In the second one add a LAYER block. Then give the layer a bottom elevation and a mixture.

```
SOIL_DEF
{
        name = fixed
}
SOIL_DEF
{
        name = mobile
        LAYER
        {
                bottom_elevation = -5.
                mixture = unique
        }
}
```

2.5.2 Soil assignment

The names of the described soils have now to be assigned to the soil codes used in the topography file. Add a SOIL_ASSIGNMENT block and there the attributes type, index and soil. The first value in the index window has to correspond to the first name in the soil window etc.



type: index_table

index: 1 2

soil:
fixed
mobile

Fig. 10: Soil assignment in the user interface

2.5.3 Define general parameters for sediment transport

The porosity and the density of the material are standard values.
The active layer is kept fixed and set to 20 cm.
The tables for the hydraulic computation will be updated each time when the bed level has changed more than 5 cm.

```
PARAMETER
{
     porosity = 0.37
     control_volume_thickness = 0.2
     density = 2650
     max_dz_table = 0.05
}
```

2.5.4 Define specific parameters for bed load transport

The Meyer-Peter and Müller bed load approach will be applied without adjusting the calculated transport capacity (bedload_factor = 1.0). The parameter for upwind scheme is set to 1 and for the critical angle a standard value has been choosen.

```
PARAMETER
{
      bedload_transport = mpm
      bedload_factor = 1
      upwind = 1
      angle_of_repose = 30
}
```

2.5.5 Define boundary conditions for bed load

At the downstream boundary it is considered that the quantity of sediment which enters the last element leaves it by the boundary.

```
BOUNDARY
{
      type = IODown
      string = downstream
}
```

At the upper boundary, the observed modification of the bed level before and after the floods is very small. For this reason it can be assumed that at the upstream boundary there is as much sediment coming in, as is transported out of the first element.

```
BOUNDARY
{
      type = IOUp
      string = upstream
}
```

2.5.6 Generate a “geometry” file

To see how the geometry of cross section 14 changes during the flood add a special output to the *OUTPUT* block.

```
OUTPUT
{
      output_time_step = 1000
      console_time_step = 1000
      SPECIAL_OUTPUT
      {
            type = monitor
            output_time_step = 1000
            cross_sections = ( CS14 )
            geometry = ( time )
```

This page has been intentionally left blank.

2.6 Perform bed load simulation (Thur 3)

Run the file Thur3.bmc.

When the simulation is terminated look at the Thur3out file, take the columns of distance and mean Bottom level of the start and end situation and make a longitudinal profile of it.

Additionally open the topology file of cross section 15 and plot the old and new geometry of the cross section.

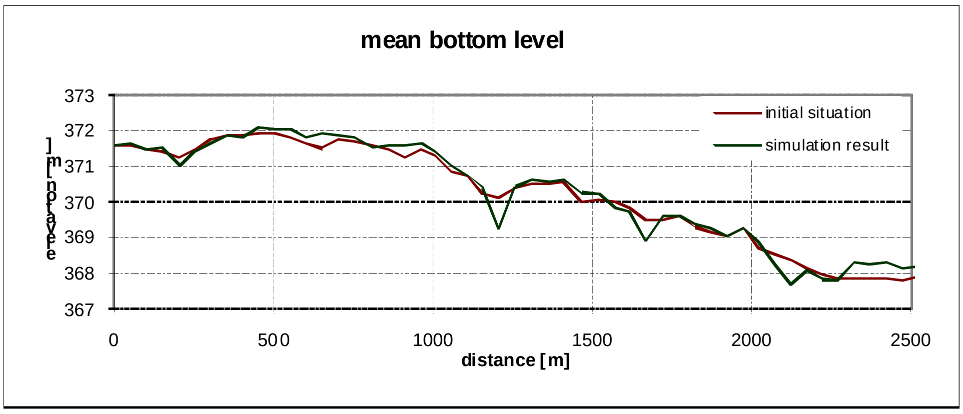


Fig. 11: Longitudinal profile of mean bottom level

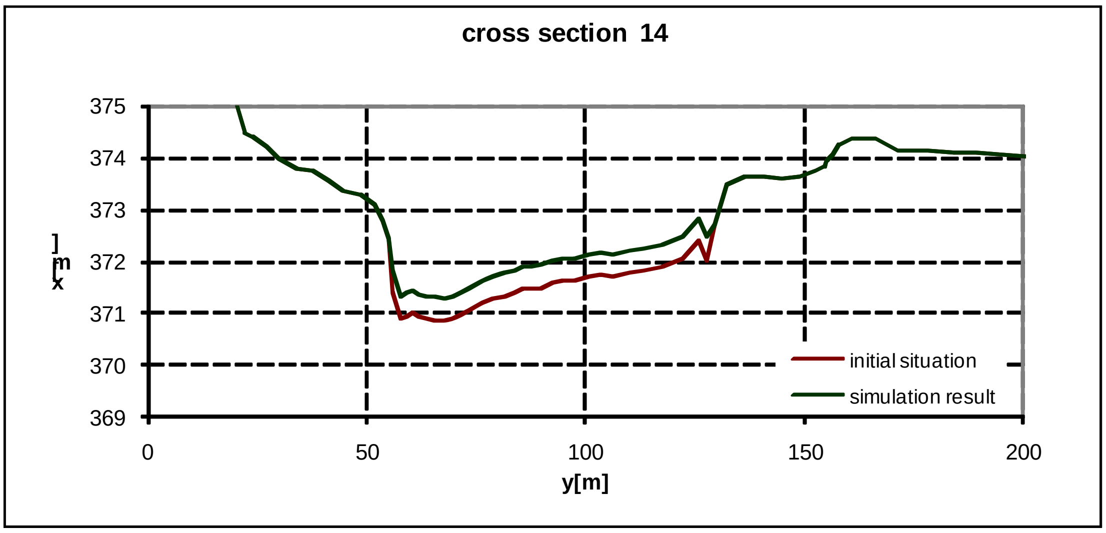


Fig. 12: Cross section

Obviously this is only a first run for exercise. This computation now needs calibration and validation before it can be used to make prediction of future evolution.

This page has been intentionally left blank.

2.7 Hydraulic computation using HEC-RAS 3.1.3

In future releases BASEMENT will only support its own topography file directly. For the support of HEC-RAS topography files, a separate converter program is planned.

This page has been intentionally left blank.

IV TUTORIALS

Part 2: Applications of BASEplane

3 Hydrodynamics and sediment transport at the river Flaz

3.1 Introduction

This tutorial gives an introduction to the capabilities of the 2-D modelling module BASEplane of BASEMENT. It provides a step-by-step guidance on how build up a model for BASEplane.

3.1.1 Case study description

The tutorial for the 2-D modelling module is based on an extract of the case study of the river Flaz in Graubünden. Within the framework of a high water protection project for the village Samedan a completely new section of the river Flaz was built. On a length of 4.1 km morphologically different kind of river subsections can be distinguished (Fig. 13). The numerical modelling of the whole domain is carried out within the river monitoring project Flaz of the Laboratory of Hydraulics, Hydrology and Glaciology (VAW).
In order to reduce the model size (and thus computational run time) only the three most interesting sub-sections are modelled in this tutorial, such as the lower part of the section enriched with roughness elements, the widening part and the part with alternating bars shown in Fig. 13.

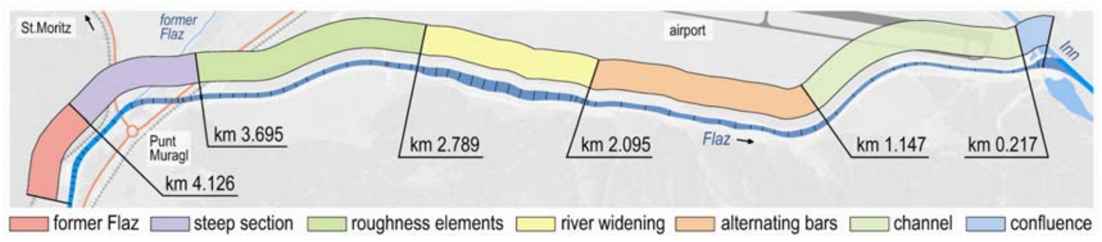


Fig. 13: Different morphological river subsections of the new section of the river Flaz.

3.1.2 Tutorial structure

In a first step the important properties of the mesh file are shown. The tutorial is designed to run BASEplane with the help of the software SMS[1] as pre- and post-processor. Furthermore the results can be post-processed (visualized) with the software Tecplot[2]. Though, the main focus is on the setup the command file for the numerical simulation with BASEMENT. The tutorial is structured gradually in the way that first of all a calibrated hydraulic model is set up. Based on this simulation the morphological part can be added to the simulation with a single-grain model. This procedure reflects the proposed way from a calibrated hydraulic simulation to a morphological simulation. An outline of outputs is given and possible visualization is shown for each step.

[1] http://www.aquaveo.com/sms

[2] http://www.tecplot.com

3.2 Computational grid

The computational mesh is generated with the pre-processor program SMS. Its detailed setup with SMS is not part of this tutorial. Here just important features and characteristics of the computational mesh for the modelling with BASEMENT are mentioned. The mesh discretizes the topography of the river in such a way that the important topographical information is maintained. Break lines in the mesh are ensuring that important features of the topography such as the river bed and dike crests are represented correctly (Fig. 13).

An important feature is the assignment of the material index to the different groups of elements. By the material index different properties such as the friction factor and the soil properties can be assigned. The material index is mainly used to assign the friction factor to the different river sections separately. For example it is usual to assign different values for the main channel, embankments and further surrounding land (Fig. 15).

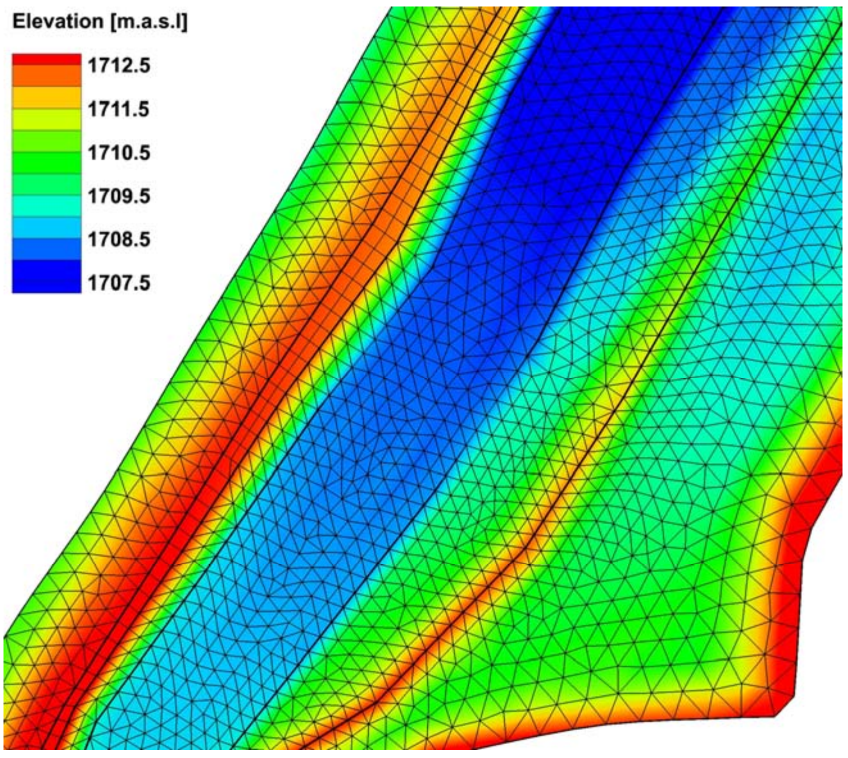


Fig. 14: Computational mesh generated by SMS for a section in the widening part. The bold black lines are break lines for the mesh.

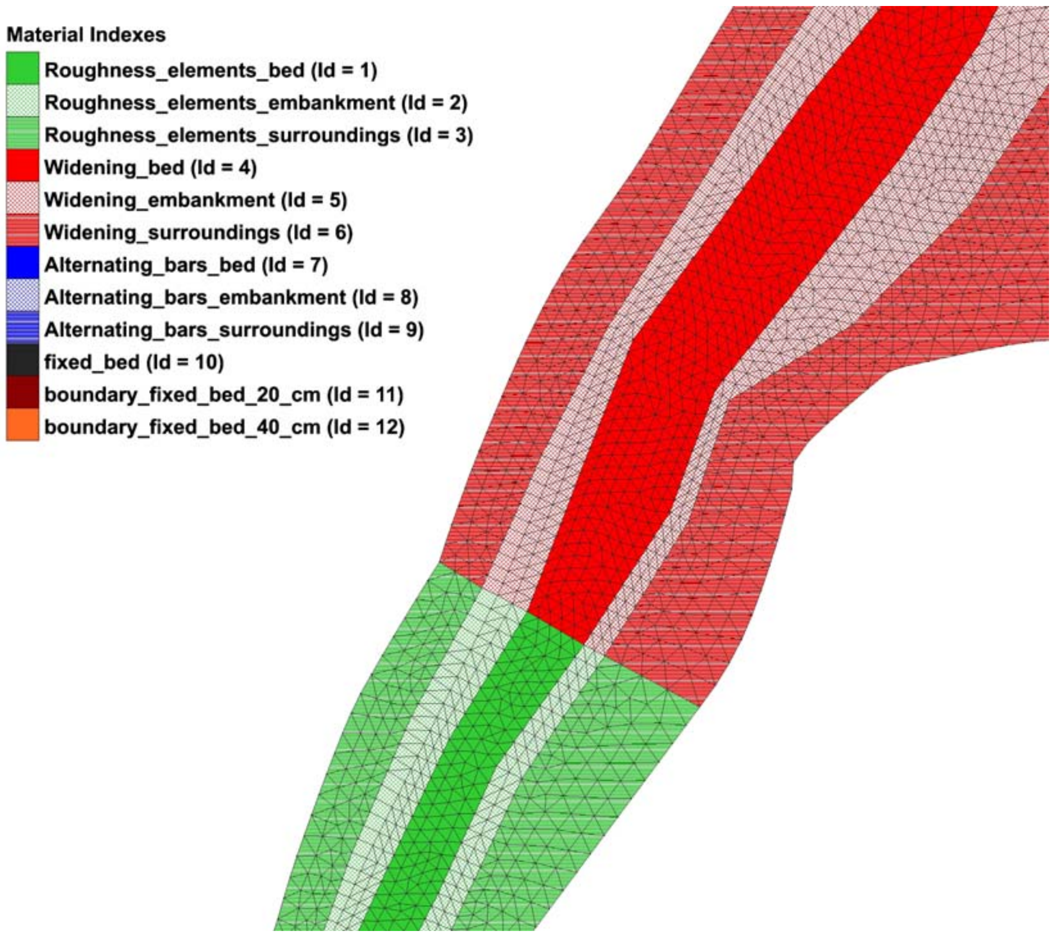

Material Indexes
Roughness_elements_bed (Id = 1)
Roughness_elements_embankment (Id = 2)
Roughness_elements_surroundings (Id = 3)
Widening_bed (Id = 4)
Widening_embankment (Id = 5)
Widening_surroundings (Id = 6)
Alternating_bars_bed (Id = 7)
Alternating_bars_embankment (Id = 8)
Alternating_bars_surroundings (Id = 9)
fixed_bed (Id = 10)
boundary_fixed_bed_20_cm (Id = 11)
boundary_fixed_bed_40_cm (Id = 12)

Fig. 15: Material indexes (Ids) used for assignment the friction factor.

The mesh file is saved as *Flaz_mesh.2dm* and has the following structure:

```
MESH2D
.
E3T    eN    n1    n2    n3      eMi           (triangle element)
.
E4Q    eN    n1    n2    n3    n4      eMi     (qudrilateral element)
.
ND      ni      x     y     z
.
```

Where E3T and E4Q are the flags for the triangular and quadrilateral elements respectively; eN denotes the element number; n1, n2, n3 and n4 denote the node numbers of the element and eMi is the material index of the element. The elements are defined in a counter-clockwise direction. The coordinates of the nodes are defined in the second block. ND is the flag for a node; ni denotes node number and x, y and z are the coordinates of the node.

3.3 Setting up the command file

The command file (with the ending .bmc) can be built up and changed within the graphical user interface (GUI) or in any text editor. It has the following general structure:

```
PROJECT
{...}
DOMAIN
{
      multiregion = Flaz
PHYSICAL_PROPERTIES
{...}
BASEPLANE_2D
{
      region_name = Flaz
      GEOMETRY
      {...}
      HYDRAULICS
      {...}
      MORPHOLOGY
      {...}
      OUTPUT
      {...}
}
}
```

3.3.1 Project

In this block the project name, the author and the date will be set.

```
PROJECT
{
      title = 2D_Tutorial
      author = LV
      date = 23.11.2010
}
```

3.3.2 Domain

The DOMAIN-block includes all necessary blocks for a simulation.

```
DOMAIN
{
     multiregion = Flaz
     PARALLEL
     {...}
     PHYISICAL_PROPERTIES
     {...}
     BASEPLANE_2D
     {...}
}
```

3.3.3 Parallel

In the PARALLEL-block the number of processors can be assigned to the computation with BASEMENT. Depending on the computer the number of threads can be adjusted.

```
PARALLEL
{
     number_threads = 2            // on dual-core system
}
```

3.3.4 Physical properties

The physical properties are global constants in a project.

```
PHYSICAL_PROPERTIES
{
     gravity = 9.81                // [m/s2]
     viscosity = 1.0e-6            // [m2/s]
     rho_fluid = 1000              // [kg/m3]
}
```

3.3.5 Two dimensional simulation

The *BASEPLAIN_2D*-block within the DOMAIN-block contains all information concerning the two dimensional simulation.

```
BASEPLANE_2D
{
      region_name = Flaz
      GEOMETRY
      {...}
      HYDRAULICS
      {...}
      MORPHOLOGY
      {...}
      OUTPUT
      {...}
}
```

3.3.5.1 Geometry

The GEOMETRY-block defines the mesh file and necessary strings of nodes. Strings are used for inflow and outflow boundaries and can also be used for discharge control. The node ids of the inflow and outflow string can be read out from the mesh in Fig. 16 and Fig. 17 respectively.

```
GEOMETRY
{
      type = sms
      file = Flaz_mesh.2dm

      STRINGDEF
      {
            name = Inflow
            node_ids = ( 1 2 3 4 5 6 7 8 9 )
      }
      STRINGDEF
      {
            name = Outflow
            node_ids = ( 9478 9479 9499 9500 9501 9519 9533 9546 9552 )
      }
}
```

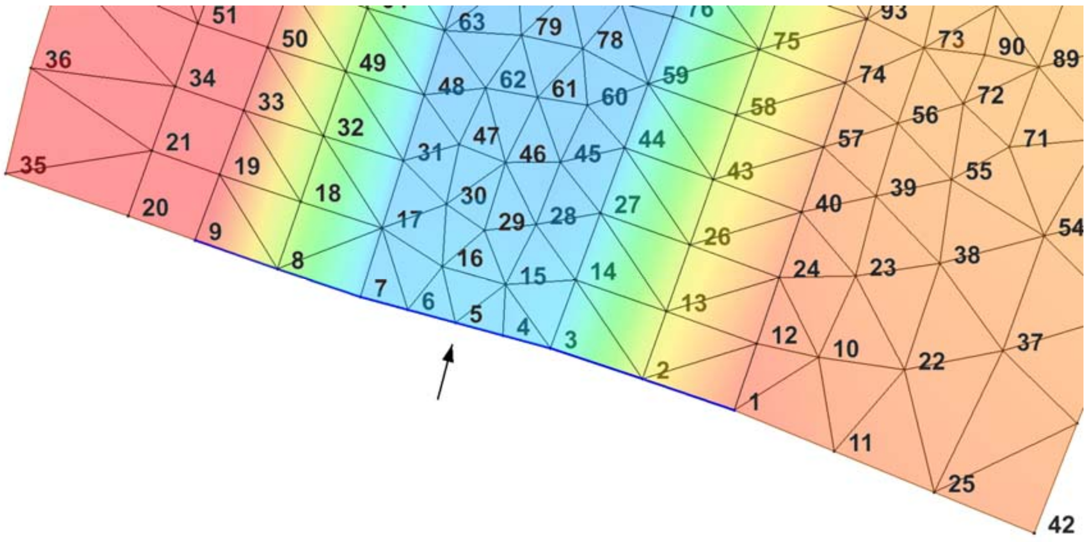


Fig. 16: Node id numbers for the definition in the STRING_DEF block for the inflow boundary.

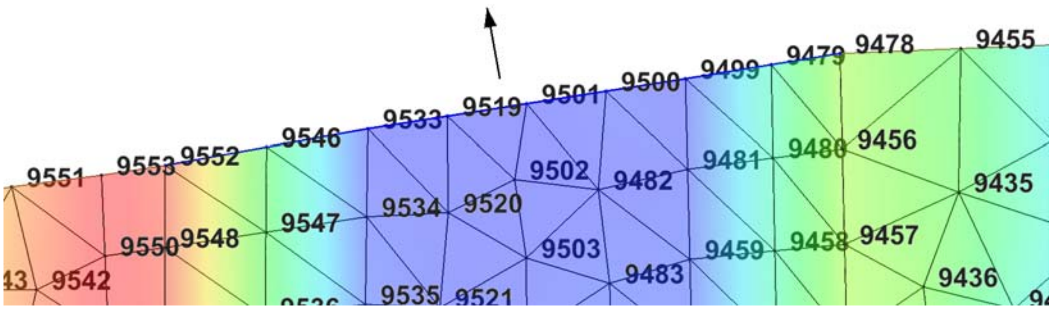


Fig. 17: Node id numbers for the definition in the STRING_DEF block for the outflow boundary.

3.3.5.1.1 Define the hydraulics

The HYDRAULIC-block includes all the information necessary for the hydraulic part of the simulation. This block is divided into the following sub-blocks:

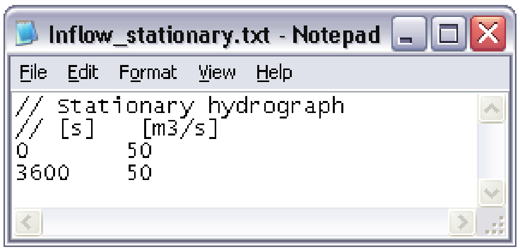
```
HYDRAULICS
{
      BOUNDARY
      {...}
      INITIAL
      {...}
      FRICTION
      {...}
      PARAMETER
      {...}
}
```

3.3.5.1.2 Hydraulic boundary conditions

For the upper (inflow) and lower (outflow) boundary condition we have to refer to the predefined STRINGDEFs (see chapter 3.3.5.1). If the boundary condition is not defined explicitly a wall boundary is considered for those edges. Except for the explicitly defined inflow and outflow boundary the model boundary is basically an impermeable wall.
The inlet boundary condition is defined across the predefined string *Inflow.* The hydraulic condition at the boundary is set by the use of a hydrograph and a corresponding slope. The normal slope is used in order to calculate the normal flow depths and the normal flow velocities at the boundary and can be considered as a calibration parameter.

```
BOUNDARY
{
      type = hydrograph
      string_name = Inflow
      file = Inflow_stationary.txt
      slope = 10.0                 // [per mill]
}
```

The hydrograph is saved in a text file *Inflow_stationary.txt* in which the first column is the time and the space separated second column is the discharge (Fig. 18). As a first step, a steady inflow hydrograph is an appropriate choice in order to test the mass conservation of the model. After a certain run time, depending on the size of the model domain, the outflow should counterbalance the inflow. There should be no uncontrolled mass loss within the model domain.
Usually the discharge is taken as the mean annual discharge or the beginning discharge of a flood event. In this case a steady discharge of 50 m^3/s is chosen in order to be able to continue later on with a flood hydrograph which starts in this range.



```
// Stationary hydrograph
// [s]    [m3/s]
0         50
3600      50
```

Fig. 18: Stationary hydrograph file saved as Inflow_stationary.txt.

The outlet boundary condition is defined across the predefined string *Outflow*. The normal slope is used in order to calculate the normal flow depths and the normal flow velocities at the boundary and can be considered as a calibration parameter. A sensitivity analysis of this parameter makes always sense. In any case the upper and lower model boundary should be far away enough from the river section of interest, in order to minimise the influence of the boundary conditions.

```
BOUNDARY
{
      type = hqrelation
      string_name = Outflow
      slope = 2.0            // [per mill]
}
```

3.3.5.1.3 Initial condition

The INITIAL-block defines the flow variables at the beginning of the simulation. In a very first step the simulation is started with a dry initial condition.

```
INITIAL
{
      type = dry
}
```

3.3.5.1.4 Friction

The FRICTION-block defines everything that is related to the friction term in the shallow water equations. Within the computational mesh a material index is assigned to all elements. By the use of this material index (see Fig. 15) a friction factor can be assigned. The default friction is used whenever there is no friction assigned to an element.

```
FRICTION
{
      type = strickler
      default_friction = 30
      input_type = index_table
      index = ( 1 2 3 4 5 6 7 8 9 10 11 12 )
      friction = ( 28 30 35 30 30 30 32 32 35 28 28 28 )
      wall_friction = off
}
```

3.3.5.1.5 Computational parameters

The PARAMETER-block defines the control parameters for the numerical simulation of the hydraulic part. The numerical simulation is performed using explicit time integration and the exact Riemann solver for flux computation. The elements with a water depth below the minimum water depth will be considered as dry elements due to stability reasons.
The simulation is performed with a total runtime of 3000 seconds. Later on, it has to be tested that after this runtime the flow in the model domain has reached a steady state, meaning that the outflow counterbalances the inflow (see next chapter).

```
PARAMETER
{
      simulation_scheme = exp
      riemann_solver = exact
      CFL = 0.95
      total_run_time = 3000
      minimum_water_depth = 0.05
      minimum_time_step = 0.0001
}
```

3.3.5.2 Define the output

In the OUTPUT-block, the desired output has to be defined. During the simulation, output can also be visualized with BASEviz.

```
OUTPUT
{
      output_time_step = 1000
      console_time_step = 100
      SPECIAL_OUTPUT
      {
            type = BASEviz
            variable = depth
            output_time_step = 10
            gridlines = off
            vectors = on
      }
      SPECIAL_OUTPUT
      {
            format = sms
            type = node_centered
            values = ( depth velocity wse )
            output_time_step = 3000
      }
      SPECIAL_OUTPUT
      {
            type = balance
            balance_values = ( timestep )
            output_time_step = 100
      }
      SPECIAL_OUTPUT
      {
            type = boundary_history
            boundary_values = ( Q )
            history_one_file = yes
            output_time_step = 100
      }
}
```

3.4 Perform hydraulic simulation

3.4.1 Perform steady flow simulation

Open the command file *Flaz_hydraulic_stationary.bmc* either by double-clicking or via the menu of the BASEMENT GUI (File -> Open Command). Run the simulation with the *Run* button of the BASEMENT window. If the SPECIAL_OUTPUT of the type *BASEviz* is chosen press the keyboard button p to start the simulation. Be aware that the mesh, command file and all other input files have to be in the same folder. The output files are stored in this same folder. In order to check the mass conservation of the model, the file *Flaz_combined_th.dat* is used. After approximately 1600 seconds the outflow counterbalances the steady inflow (Fig. 19).

In the file *Flaz_balance.dat* the run time of the simulation, the computational time steps and the element which is limiting the computational time step are stored. The identification of the limiting element allows for improvement of the mesh at the indicated location by the use of the mesh generator.

The solution files with the ending *.sol* can be imported into the program SMS and the water depth and flow velocities can be visualized as shown in Fig. 20. At the end of the simulation the flow variables of the last time step (t = 3000 s) are stored in the *Flaz_old_Flaz.sim* file. This file can be used later on to continue the simulation.

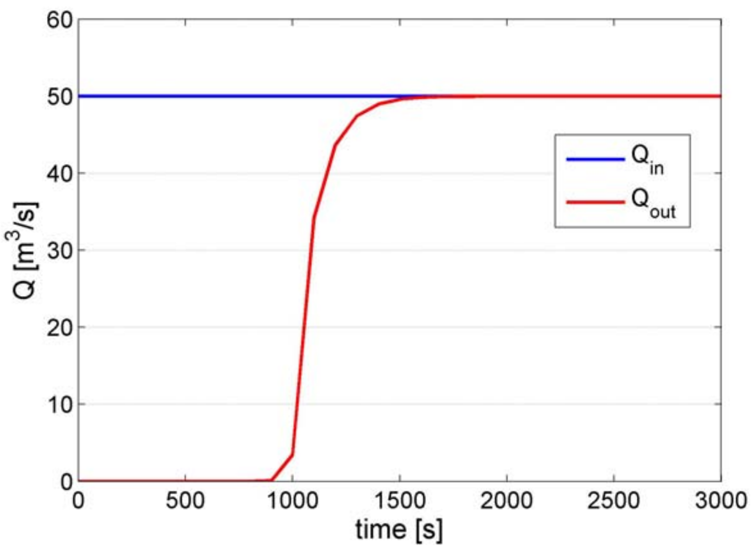


Fig. 19: Steady inflow hydrograph and outflow hydrograph.

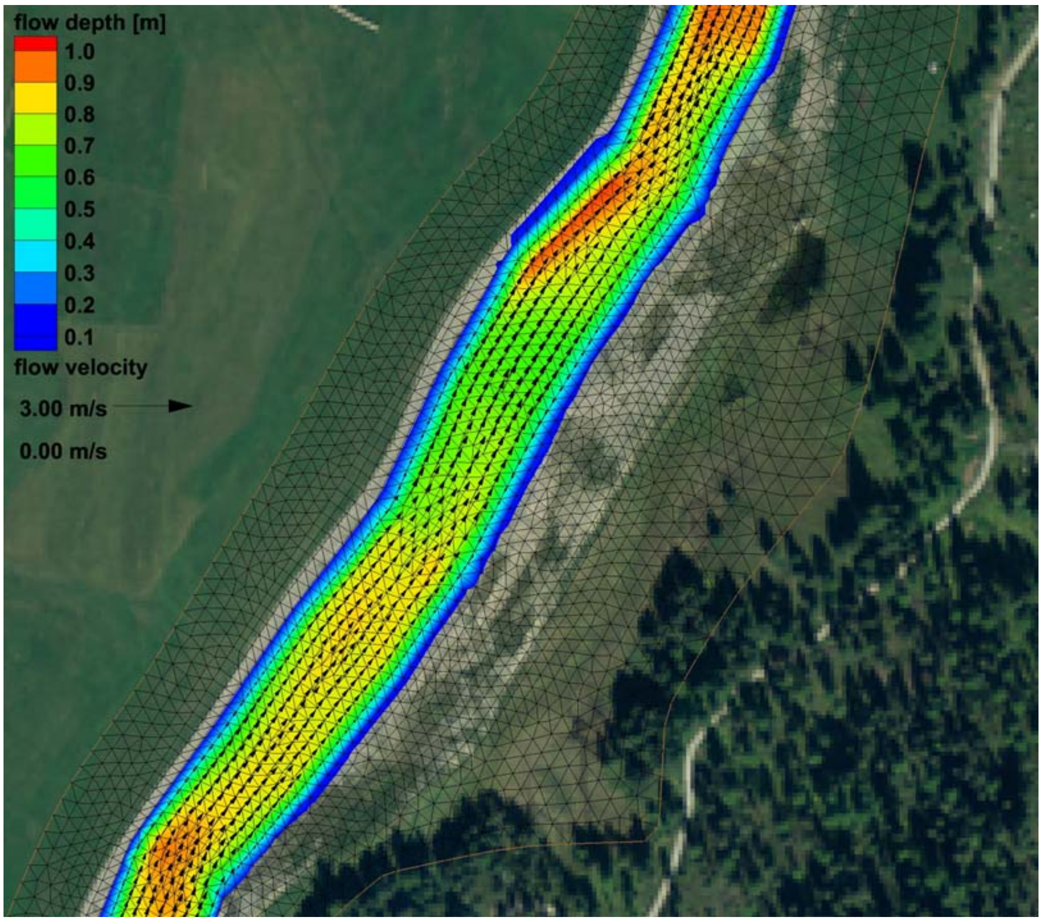


Fig. 20: Flow depth and flow velocity vectors at the steady state of the model.

3.4.2 Perform unsteady flow simulation

The unsteady flow simulation is based on the flood event of July 2004 depicted in Fig. 21. Compared to the steady flow simulation the command file needs some minor changes. First of all the last time step of the steady simulation is taken as initial condition for the unsteady simulation. Therefore the file *Flaz_restart.cgns* from the steady simulation can be renamed and saved for example as *Initial_Condition.cgns*. This file now can be used as initial condition for the unsteady flow simulation as follows:

```
INITIAL
{
      type = continue
      file = Initial_Condition.cgns
      restart_solution_time = 3000.027
      restart_start_time = 0
}
```

With the tag restart_solution_time the solution of the last time step of the stationary simulation is chosen. In order to start the simulation from the beginning, the restart_start_time is set to zero.
Furthermore the inflow hydrograph of the flood has to be defined and assigned to the upper boundary condition. The hydrograph of the flood shown in Fig. 21 is saved in the text file *Inflow_instationary.txt* with contents as depicted in Fig. 22. Be aware that the final time defined in this file has to be the same or larger than the computation time. The upper BOUNDARY-block changes to:

```
BOUNDARY
{
     type = hydrograph
     string_name = Inflow
     file = Inflow_instationary.txt
     slope = 10.0    // [per mill]
}
```

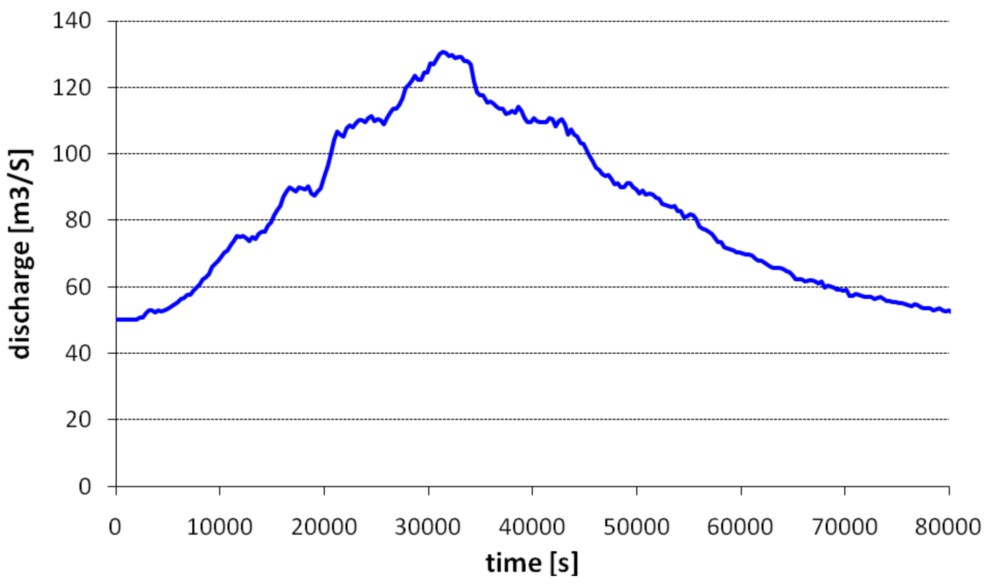
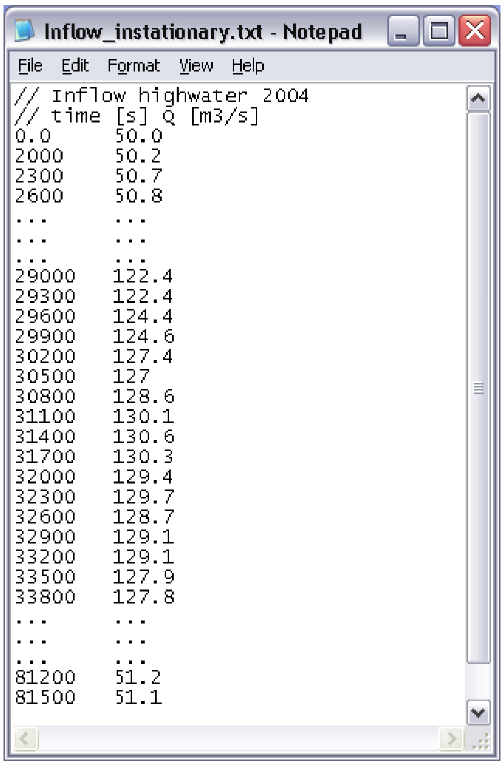


Fig. 21: Hydrograph of the flood event of July 2004.



```
// Inflow highwater 2004
// time [s] Q [m3/s]
0.0      50.0
2000     50.2
2300     50.7
2600     50.8
...      ...
...      ...
...      ...
29000    122.4
29300    122.4
29600    124.4
29900    124.6
30200    127.4
30500    127
30800    128.6
31100    130.1
31400    130.6
31700    130.3
32000    129.4
32300    129.7
32600    128.7
32900    129.1
33200    129.1
33500    127.9
33800    127.8
...      ...
...      ...
...      ...
81200    51.2
81500    51.1
```

Fig. 22: Inflow hydrograph stored in the file Inflow_instationary.txt. Note that the points (...) are just illustrative in order to show the first and last line of the file.

In the PARAMETER-block the total run time of the simulation is increased up to 80'000 seconds in order to capture the whole flood event:

```
PARAMETER
{
...
      total_run_time = 80000
...
}
```

Last but not least the OUPUT-block has to be adjusted to the needs of the simulation. For the unsteady simulation maxima values may be of interest.

```
OUTPUT
{
     output_time_step = 2000
     console_time_step = 100
     SPECIAL_OUTPUT
     {
          format = sms
          type = node_centered
          values = ( depth velocity wse max[depth] max[velocity]
                    max[wse] )
          output_time_step = 10000
     }
     SPECIAL_OUTPUT
     {
          format = tecplot
          binary = yes
          type = element_centered
          values = ( max[depth] max[wse] max[velocity] )
          output_time_step = 80000
     }
     SPECIAL_OUTPUT
     {
          type = balance
          balance_values = ( timestep )
          output_time_step = 500
     }
     SPECIAL_OUTPUT
     {
          type = boundary_history
          boundary_values = ( Q )
          history_one_file = yes
          output_time_step = 500
     }
}
```

Open the command file *Flaz_hydraulic_instationary.bmc* either by double-clicking or via the menu of the BASEMENT GUI (File -> Open Command). Run the simulation with the *Run* button of the BASEMENT window.
The maxima values of the flow depths and flow velocity vectors can be visualized using SMS as shown in Fig. 23.

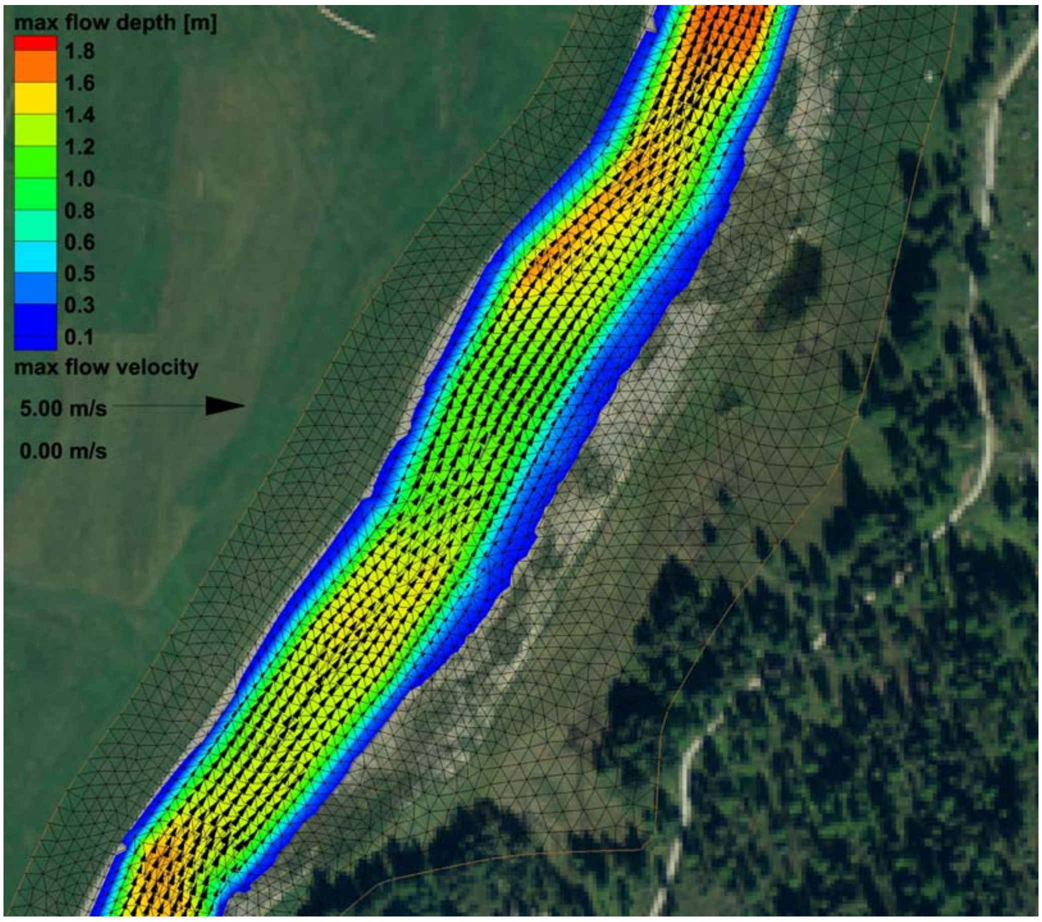


Fig. 23: Maximal flow depth and flow velocity vectors of the unsteady flow simulation.

3.4.3 Calibration of the hydraulic model

The hydraulic model can be calibrated for example based on flood level marks by comparing the modelled water surface elevations with the flood level marks. Usually the calibration parameter is the bed roughness introduced with the Strickler value. The calibration procedure may need several adjustments and is an iterative process. The demonstration of the calibration is not part of this tutorial. It should be mentioned that it is important to have a calibrated hydraulic model either for further hydraulic modelling or for morphological modelling in a further step.

3.5 Morphological simulation with single-grain bed load transport

The MORPHOLOGY-block is not compulsory. If this block is not defined the simulation is purely hydraulic. The command file of the unsteady hydraulic simulation has to be completed for the single-grain bed load transport as shown in this section.
The morphological simulation is based on the flood event in July 2004. Therefore a single-grain bed load transport is added to the unsteady hydraulic simulation in chapter 3.4.2. In the HYDRAULIC-block a small change has to be done in order to define the boundary string *Inflow_sed* for the bed load inflow. Thus a new STRINGDEF-block is added within the GEOMETRY-block as follows:

```
GEOMETRY
{
      ...
      STRINGDEF
      {
            name = Inflow_sed
            node_ids = ( 3 4 5 6 7 )
      }
      ...
}
```

3.5.1 Define the morphological information

The necessary information for the morphological part of the simulation is defined in the *MORPHOLOGY*-block.

```
MORPHOLOGY
{
      PARAMETER
      {...}
      BEDMATERIAL
      {...}
      BEDLOAD
      {...}
      GRAVITATIONAL_TRANSPORT
      {...}
}
```

3.5.1.1 Morphological parameters

In the PARAMETER-block important parameters for the morphological simulation are defined. The bed load control volume is chosen to be constant with a thickness of 0.1 m.

```
PARAMETER
{
     porosity =  40                      // [%]
     density = 2650                      // [kg/m3]
     control_volume_type = constant
     control_volume_thickness = 0.1      // [m]
}
```

3.5.1.2 Bed material

In the BEDMATERIAL-block the grain classes, the composition, the thickness of the soil layers, the level of the fixed bed and the assignment of the soil to the mesh is defined in several sub-blocks.

```
BEDMATERIAL
{
     GRAIN_CLASS
     {...}
     MIXTURE
     {...}
     SOIL_DEF
     {...}
     FIXED_BED
     {...}
     SOIL_ASSIGNMENT
     {...}
}
```

3.5.1.3 Grain size distribution

The single-grain simulation is performed with only one grain class of a given diameter, e.g. the mean grain diameter.

```
GRAIN_CLASS
{
     diameters = ( 50 )  // [mm]
}
```

3.5.1.4 Grain mixture

Since we have only one grain size, the volume fraction is equal to 100%.

```
MIXTURE
{
      name = single_grain
      volume_fraction = ( 100 )    // [%]
}
```

3.5.1.5 Define the soil composition

The soil layers and the according sediment mixture are defined in the SOIL_DEF-block. For a single-grain simulation it is not important how many layers are defined. The negative bottom elevation defines the thickness of the layer. Below the last layer a fixed bed is assumed. If no LAYER-block is defined, then automatically a fixed bed on the surface is assumed. We use this especially for the river bed near the upper boundary condition to avoid uncontrolled erosion. Furthermore the embankments are kept fixed because the main focus is on the river bed morphology. The two soils *soil_fix_20* and *soil_fix_40* are defined to have a gradual transition from the fixed bed to the movable bed. Anyway, the river section with the roughness elements cannot be modelled accurately, because single roughness elements which are more or less fixed stones cannot be discretized within the computational mesh. They have to be modelled with an increased bed roughness instead.

```
SOIL_DEF
{
      name = soil_roughness_elements
      LAYER
      {
      bottom_elevation = -0.8      // fixed bed 0.8 m below the surface
      mixture = single_grain
      }
}
SOIL_DEF
{
      name = soil_widening
      LAYER
      {
      bottom_elevation = -2.0      // fixed bed 2.0 m below the surface
      mixture = single_grain
      }
}
```

```
    SOIL_DEF
    {
        name = soil_alt_bars
        LAYER
        {
        bottom_elevation = -2.0      // fixed bed 2.0 m below the surface
        mixture = single_grain
        }
    }
    SOIL_DEF
    {
        name = soil_fix_20
        LAYER
        {
        bottom_elevation = -0.2      // fixed bed 0.2 m below the surface
        mixture = single_grain
        }
    }
    SOIL_DEF
    {
        name = soil_fix_40
        LAYER
        {
        bottom_elevation = -0.4      // fixed bed 0.4 m below the surface
        mixture = single_grain
        }
    }
    SOIL_DEF
    {
        name = soil_fix              // fixed bed
    }
```

3.5.1.6 Fixed bed elevation

There are several possibilities to define a fixed bed. In the FIXED_BED-block, the elevations of areas with fixed bed can be defined either with a separate mesh file containing the fixed bed elevations or with specific fixed bed elevations for some selected nodes. Furthermore a fixed bed can be implemented in the SOIL_DEF-block (chapter 3.5.1.5). If there is no layer defined, a fixed bed will be assumed. In any case, a fixed bed is assumed below the last layer. In this tutorial the FIXED_BED-block is used as an example to define a fixed bed for a single node. This can be used to consider a big stone for example. A fixed node (node id 8956) is implemented by giving *zb_fix* a value smaller or equal to -100.

```
FIXED_BED
{
      type = nodes
      node_ids = ( 8956 )
      zb_fix = ( -100 )
}
```

3.5.1.7 Assignment of the defined soil types

The soil types defined in the SOIL_DEF-blocks (section 3.5.1.5) are assigned to the elements of the mesh by the material index.

```
SOIL_ASSIGNMENT
{
      type = index_table
      index = ( 1  2  3  4  5  6  7  8  9  10  11  12 )
      soil = ( soil_roughness_elements  soil_fix  soil_fix
                  soil_widening  soil_fix  soil_widening  soil_alt_bars
                  soil_fix  soil_fix  soil_fix  soil_fix_20  soil_fix_40
)
}
```

3.5.1.8 Initial condition

The initial bed elevation is defined in most cases as the actual topography. If there is no INITIAL-block defined, the initial bed elevation will be automatically set to the bed elevation in the computational mesh file.

3.5.1.9 Bed load boundary condition

The BOUNDARY-block within the BEDLOAD-block defines the boundary condition for the bed load transport. The bed load input is handled with a boundary condition which determines the transport capacity at the inflow cross section. The *IODown* is the only downstream boundary condition available for sediment transport at the moment. All sediment entering the last computational cell will leave the cell over the downstream boundary.

```
BOUNDARY
{
      type = transport_capacity
      string_name = Inflow_sed
      mixture = single_grain
      factor = 1.0
}
```

```
BOUNDARY
{
      type = IODown
      string_name = Outflow
}
```

3.5.1.10 Bed load parameter

The control parameters for the bed load simulation are defined in the PARAMETER-block within the BEDLOAD-block. The bed load transport is computed with the Meyer-Peter and Mueller’s (mpm) formula. Since the *limit_bedload_wetted* tag is turned off, the bed load is computed not only in completely wetted cells but in partially wetted cells as well. The lateral transport caused by transversal slope in relation to the flow velocity is taken into account.

```
PARAMETER
{
      bedload_factor = 0.5
      bedload_transport = mpm
      theta_critic = ( 0.04 )
      limit_bedload_wetted = off
      lateral_transport = on
}
```

3.5.1.11 Gravitational transport

In the GRAVITATIONAL_TRANSPORT-block the parameters for gravitation induced transport are defined. The gravitational transport can be limited to elements which are fully wetted or can be considered for all elements. Over the material index the scope and the applied angles for the gravitational transport can be defined. Note that for soils with a fixed bed the gravitational transport is not active. The applied sediment transport formulas are basically just valid for the equilibrium sediment transport. With the gravitational transport unevenness occurring due to sediment transport will be smoothed out.

```
GRAVITATIONAL_TRANSPORT
{
      index = ( 1 2 3 4 5 6 7 8 9 10 11 12 )
      angle_failure_original_dry = ( 30 30 30 30 30 30 30 30 30 30 30
                                     30 )
      angle_failure_original_wetted = ( 15 15 15 15 15 15 15 15 15 15
                                     15 15 )
      angle_failure_deposited = ( 10 10 10 10 10 10 10 10 10 10 10 10 )
      gravity_transport_on_cells = partially wetted
      angle_wetted_criterion = partially_wetted
}
```

3.5.2 Define the output

The desired output of the simulation has to be defined explicitly in the OUTPUT-block. The *output_time_step* defines the time steps of the results. The *console_time_step* defines the time step to appear in the BASEMENT window during simulation.
Specific output modes have to be defined in the repeatable SPECIAL_OUTPUT-blocks. Inside this block the output_time_step defines the output time step for this particular output. A detailed overview of all possible output types, values, format types and more is given in the "Reference Manual IV - RIV" (Command and input files).

```
OUTPUT
{
      output_time_step = 2000
      console_time_step = 100
      SPECIAL_OUTPUT
      {
            format = sms
            type = node_centered
            values = ( depth wse velocity deltaz z_node )
            output_time_step = 20000
      }
      SPECIAL_OUTPUT
      {
            format = tecplot
            binary = yes
            type = element_centered
            values = ( z_element deltaz )
            output_time_step = 80000
      }
      SPECIAL_OUTPUT
      {
            type = balance
            balance_values = ( sediment timestep )
            output_time_step = 1000
      }
      SPECIAL_OUTPUT
      {
            type = boundary_history
            boundary_values = ( Q Qsed )
            history_one_file = yes
            output_time_step = 1000
      }
}
```

3.6 Perform morphological simulation with single-grain bed load transport

Open the command file *Flaz_morphology_single_grain.bmc* either by double-clicking or via the menu of the BASEMENT GUI (File -> Open Command). Run the simulation with the Run button of the BASEMENT window. Be aware that the mesh, command file and all other input files have to be in the same folder. The defined outputs are now generated in the same folder as the command file.

The output files with the ending *.sol* can be visualized using the program SMS. The bed elevation after the flood event is shown in Fig. 24. Two cross sections are defined (Fig. 24) and the bed elevation before and after the flood event are compared (Fig. 25). The result file in the *tecplot* format can be visualized by the use of the program Tecplot. As an example the morphological changes (*deltaz)* are shown in Fig. 26.

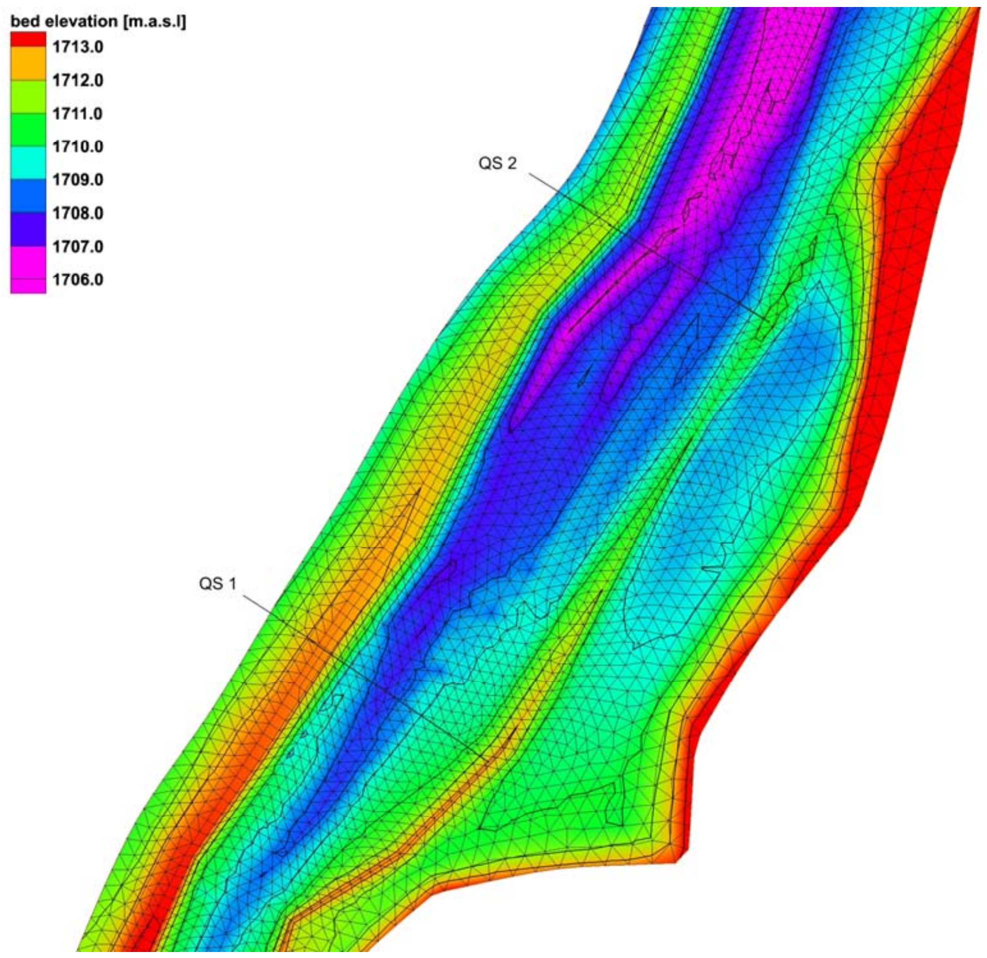


Fig. 24: Modeled bed elevation (z_bed) and two cross sections defined in the widening part.

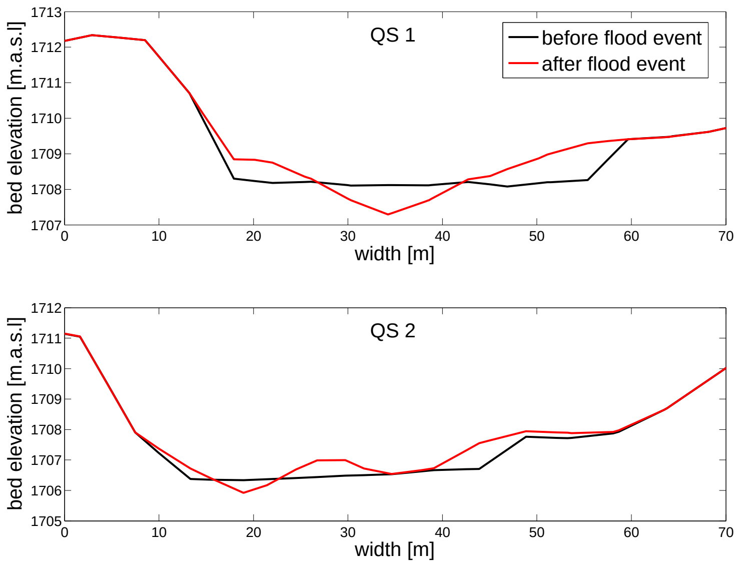


Fig. 25: Comparison of the river bed before and after the flood event in cross-section QS 1 and QS 2.

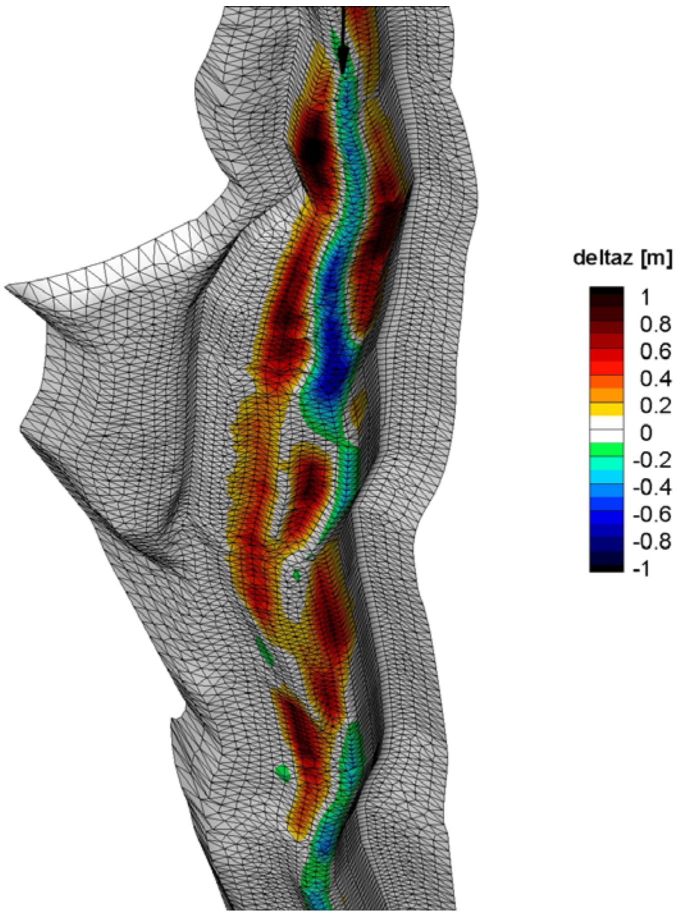


Fig. 26: Changes of the morphology (deltaz) due to the flood event with the single-grain model. The red colour range represents deposition and the blue colour range shows erosion.

3.7 Morphological simulation with multi-grain bed load transport

In order to avoid needless duplication compared to the single-grain simulation just the modifications of the command file are pointed out.
Basically there is the possibility to use the grain size distribution to determine the bed friction in the FRICTION-Block. To simplify matters the friction is defined with the Strickler value. Generally it is suggested to try both options and to choose the most suitable for your model purpose.

3.7.1.1 Morphological parameters

In the PARAMETER-block important parameters for the morphological simulation are defined. In multi-grain simulations the thickness of the bed load control volume is an important calibration parameter. This parameter influences significantly the grain sorting process.

```
PARAMETER
{
      ...
      control_volume_type = constant
      control_volume_thickness = 0.1      // [m]
}
```

3.7.1.2 Grain size distribution

The grain size distribution is discretized with six grain classes. They have to be defined in ascending order from the smallest to the largest grain.

```
GRAIN_CLASS
{
      diameters = ( 1 5 15 44 82 150 )    // [mm]
}
```

3.7.1.3 Grain mixture

In the MIXTURE-block the volume fraction of the different mixtures are defined. The three river sections are considered with different sediment mixtures. Furthermore a mixture for the inflow is defined.

```
MIXTURE
{
      name = mixture_inflow
      volume_fraction = ( 15 15 23 27 10 10 )
}
MIXTURE
{
      name = mixture_roughness_elements
      volume_fraction = ( 17 11 14 27 14 17 )
}
MIXTURE
{
      name = mixture_widening
      volume_fraction = ( 21 13 16 25 11 14 )
}
MIXTURE
{
      name = mixture_alt_bars
      volume_fraction = ( 27 14 14 20 14 11 )
}
```

3.7.1.4 Define the soil composition

The soil layers with the corresponding sediment mixture are defined in the SOIL_DEF-block. The soil can be defined with several layers of different material, but to keep it simple we assume a single layer. The negative bottom elevation defines the thickness of the layer. Below the last layer a fixed bed is assumed. If no LAYER-block is defined then automatically a fixed bed on the surface is assumed. We use this especially for the river bed near the upper boundary condition to avoid uncontrolled erosion. Furthermore the embankments are kept fix because the main focus is on set on the river bed morphology. The two soils *soil_fix_20* and *soil_fix_40* are defined to have a gradual transition from the fixed bed to the movable bed.

```
SOIL_DEF
{
      name = soil_roughness_elements
      LAYER
      {
      bottom_elevation = -0.8      // fixed bed 0.8 m below the surface
      mixture = mixture_roughness_elements
      }
}
SOIL_DEF
{
      name = soil_widening
      LAYER
      {
      bottom_elevation = -2.0      // fixed bed 2.0 m below the surface
      mixture = mixture_widening
      }
}
SOIL_DEF
{
      name = soil_alt_bars
      LAYER
      {
      bottom_elevation = -2.0      // fixed bed 2.0 m below the surface
      mixture = mixture_alt_bars
      }
}
SOIL_DEF
{
      name = soil_fix      // fixed bed
}
SOIL_DEF
{
      name = soil_fix_20
      LAYER
      {
      bottom_elevation = -0.2      // fixed bed 0.2 m below the surface
      mixture = mixture_roughness_elements
      }
}
```

```
SOIL_DEF
{
      name = soil_fix_40
      LAYER
      {
      bottom_elevation = -0.4      // fixed bed 0.4 m below the surface
      mixture = mixture_roughness_elements
      }
}
```

3.7.1.5 Bed load boundary condition

The bed load input is regulated with a boundary condition which determines the transport capacity at the cross section defined. The factor for the bed load at the boundary is an important calibration parameter and depends on the transport formula. Therefore this factor is different for single-grain and multi-grain simulations. The outflow boundary is handled as in the single-grain simulation.

```
BOUNDARY
{
      type = transport_capacity
      string_name = Inflow_sed
      mixture = mixture_inflow
      factor = 1.0
}
BOUNDARY
{
      type = IODown
      string_name = Outflow
}
```

3.7.1.6 Bed load parameter

For the sediment transport computation different bed load transport formulas are available. In this tutorial the formula of Meyer-Peter and Mueller for multiple grain classes is chosen. It is suggested to try different sediment transport formulas.

```
PARAMETER
{
      bedload_transport = mpm_multi
      bedload_factor = 0.5
      limit_bedload_wetted = off
      lateral_transport = on
      lateral_transport_factor = 1.0
}
```

3.7.1.7 Define the output

The desired output of the simulation has to be defined explicitly in the OUTPUT-block. The specific output is defined in the repeatable SPECIAL_OUTPUT-blocks. For the multi-grain simulation some additional output may be interesting such as for example the grain size distribution in selected nodes. This way grain sorting effects can be observed. A detailed overview of all possible output types, values, format types and more is given in help buttons in the Command File Editor of BASEMENT.

```
OUTPUT
{
      output_time_step = 2000
      console_time_step = 100
      SPECIAL_OUTPUT
      {
            format = sms
            type = node_centered
            values = ( depth deltaz z_node )
            output_time_step = 10000
      }
      SPECIAL_OUTPUT
      {
            format = tecplot
            binary = yes
            type = element_centered
            values = ( z_element deltaz )
            output_time_step = 80000
      }
```

```
        SPECIAL_OUTPUT
        {
              type = balance
              balance_values = ( sediment timestep )
              output_time_step = 5000
        }
        SPECIAL_OUTPUT
        {
              type = node_history
              node_values = ( grain_size )
              node_ids = ( 814 4758 9312 )
              history_one_file = yes
              output_time_step = 5000
        }
        SPECIAL_OUTPUT
        {
              type = boundary_history
              boundary_values = ( Q Qsed )
              history_one_file = yes
              output_time_step = 1000
        }
  }
```

3.8 Perform morphological simulation with multi-grain bed load transport

Open the command file *Flaz_morphology_multi_grain.bmc* either by double-clicking or via the menu in BASEMENT (File -> Open Command). Run the simulation with the Run button in the BASEMENT window. Be aware that the mesh, command file and all other input files have to be in the same folder. The defined outputs are now stored in the same folder as the command file.

The morphological changes *deltaz* are shown in Fig. 27. Here the multi-grain model is not compared quantitatively with the single-grain model. Nevertheless the qualitative comparison is indicating a quite similar behaviour (Fig. 26 and Fig. 27). At this state much more details could be investigated such as the grain class fractions, the hiding-and-exposure function (*hiding_exponent*), the amount of grain classes etc. Further important calibration parameters are the critical dimensional shear stress, the bed load factor and the bed load inflow controlled with the bed load factor at the boundary.

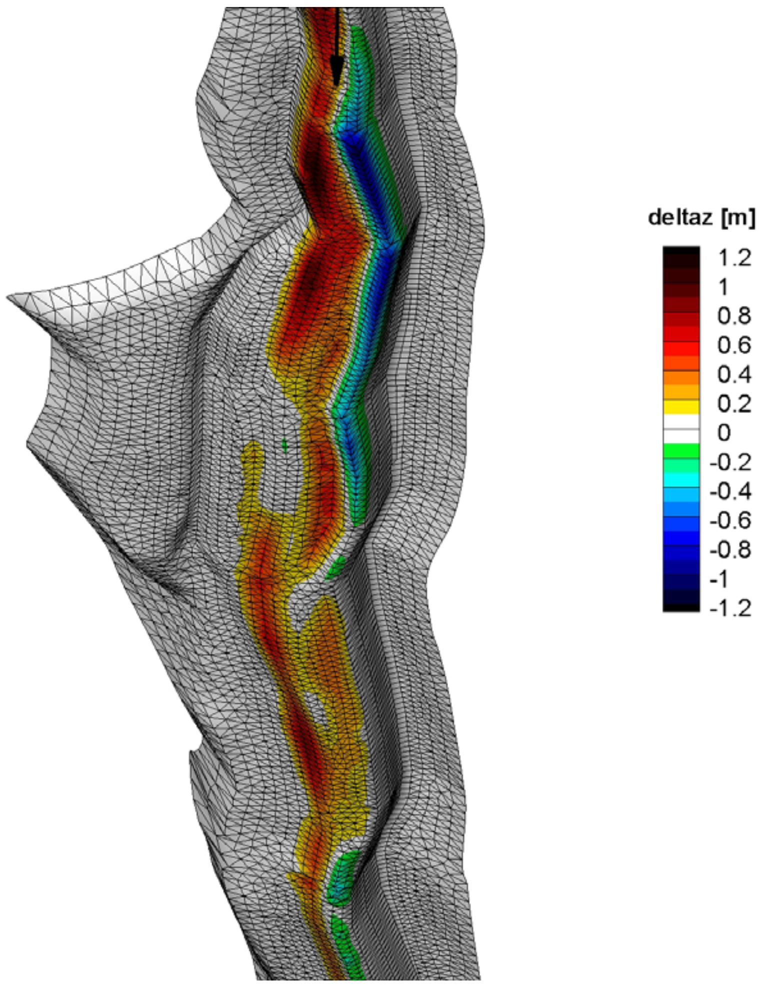


Fig. 27: Changes of the morphology (deltaz) due to the flood event with the multi-grain model. The red colour range represents deposition and the blue colour range shows erosion

This page has been intentionally left blank.

APPENDIX AND INDEX

of ***BASEMENT***

This page has been intentionally left blank.

Table of Contents

A. Notation

A.1. Super- and Subscripts A.1-1
A.2. Differential Operators A.2-1
A.3. English Symbols A.3-1
A.4. Greek Symbols A.4-1

This page has been intentionally left blank.

A. Notation

A.1. Super- and Subscripts

$(.)_B$	Property of top most soil layer for bed load (active layer)
$(.)_{cr}$	Critical value
$(.)_i$, $(.)_j$, $(.)_k$	Index corresponding to three dimensional Cartesian coordinate system $\mathbb{R}^3$ with coordinates (x, y, z)
$(.)_g$	Property corresponding to g^{th} grain size class
$(.)_L$	Property on the left hand side
$(.)_l$	Lateral property
$(.)^n$	n^{th} step of time integration
$(.)_R$	Property on the right hand side
$(.)_S$	Property at water surface
$(.)_{Sub}$	Property of bed material storage layer (sub layer)
$(.)_x$, $(.)_y$, $(.)_z$	Property corresponding to three dimensional Cartesian coordinate system $\mathbb{R}^3$ with coordinates (x, y, z)
$(.)_{(.),(.)}$	Dual property, i.e. $Q_{B,x}$ = bed load discharge in x direction
$(.)_{(.)_{(.)},(.)}$	Triple property, i.e. $Q_{B_g,x}$ = bed load discharge of grain size class g in x direction

This page has been intentionally left blank.

A.2. Differential Operators

$\frac{\mathrm{d}}{\mathrm{d}x}$	Differential operator for derivation with respect to variable x
$\frac{\mathrm{d}^n}{\mathrm{d}x^n}$	Differential operator for derivation of order n w. r. to var. x
$\frac{\partial}{\partial x}$	Partial differential operator for derivation w. r. to variable x
$\frac{\partial^n}{\partial x^n}$	Partial differential operator for derivation of order n w. r. to var. x
∇	Nabla operator. In three-dimensional Cartesian coordinate system $\mathbb{R}^3$ with coordinates (x, y, z): $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)^T$

This page has been intentionally left blank.

A.3. English Symbols

Symbol	*Unit*	*Definition*
A	[m^2]	Wetted cross section area
$\mathbf{a}$	[m/s^2]	Acceleration
A_{red}	[m^2]	Reduced area
c	[m/s]	Wave speed
c_f	[-]	Friction coefficient
C	[-]	Concentration
c_μ	[-]	Dimensionless coefficient (used for turb. kin. viscosity)
d_g	[m]	Mean grain size of the size class g
d_m	[m]	Arithmetic mean grain size according to Meyer-Peter & Müller
$\mathbf{F}(\mathbf{U}),\mathbf{G}(\mathbf{U})$	[-]	Flux vectors
$\mathbf{F}$	[N]	Force
$\mathbf{g}$	[m/s^2]	Gravity
h	[m]	Water depth, flow depth
h_B	[m]	Thickness of active layer
K	[m^3/s]	Conveyance factor $K = k_{st} A R^{2/3}$
k_{st}	[m$^{1/3}$/s]	Strickler factor
k_s	[mm]	equivalent roughness height
m	[kg]	Mass
$\mathbf{n}$	[m/s]	Normal (directed outward) unit flow vector of a computational cell
ng	[-]	Total number of grain size classes
M	[Ns]	Momentum
P	[N/m^2]	Pressure
P	[m]	Hydraulic Perimeter
p	[-]	Porosity
p_B	[-]	Porosity of bed material in active layer
p_{Sub}	[-]	Porosity of bed material in sub layer
Q	[m^3/s]	Stream- or surface discharge
Q_B	[m^3/s]	total bed load flux for cross section
q_{B_g}	[m^3/s/m]	total bed load flux of grain size class g per unit width
$q_{B_g,x}$, $q_{B_g,y}$	[m^3/s/m]	Cartesian components of total bed load flux q_{B_g}
$q_{B_g,xx}$, $q_{B_g,yy}$	[m^3/s/m]	Cartesian comp. of bed load flux due to stream forces
$q_{B_g,xy}$, $q_{B_g,yx}$	[m^3/s/m]	Cartesian comp. of lateral bed load flux
q_l	[m^2/s]	Specific lateral discharge (discharge per meter of length)
R	[m]	Hydraulic radius
$\mathbf{S}(\mathbf{U})$	[-]	Vector of source terms
S_f	[-]	Friction slope
S_B	[-]	Bed slope
S_g	[m/s]	suspended load source per cell and grain size class
Sf_g	[m/s]	active layer floor source per cell and grain size class

Sl_g	[m/s]	local sediment source per cell and grain size class
t	[s]	Time
$\mathbf{U}$	[-]	Vector of conserved variables
$\mathbf{u}$	[m/s]	Flow velocity vector with Cartesian components (u,v,w)
u_*	[m/s]	shear stress veleocity
u,v,w	[m/s]	Cartesian components of flow velocity vector $\mathbf{u}$
$\overline{u},\overline{v}$	[m/s]	Cartesian components of depth averaged flow velocity
u_B,v_B,w_B	[m/s]	Cartesian components of flow velocity at bottom
u_S,v_S,w_S	[m/s]	Cartesian components of flow velocity at water surface
V	[m^3]	Volume
x,y,z	[-]	Cartesian coordinate axes
x,y,z	[m]	Distance in corresponding Cartesian direction
z_B	[m]	Bottom elevation
z_S	[m]	Water surface elevation
z_F	[m]	Elevation of active layer floor

A.4. Greek Symbols

Symbol	*Unit*	*Definition*
α_B	[-]	Empirical parameter depending on the dimensionless Shear stress of the mixture
β_g	[-]	volumetric fraction of grain size class g in active layer
$\beta_{g,Sub}$	[-]	volumetric fraction of grain size class g in active layer
Γ	[-]	Eddy diffusivity
κ	[-]	Kármán constant
Δt	[s]	Computational time step
Δt_h	[s]	Time step for hydraulic sequence
Δt_s	[s]	Time step for sediment transport sequence
Δt_{seq}	[s]	Overall, sequential time step
$\Delta x, \Delta y, \Delta z$	[m]	Grid spacing according to three dimensional Cartesian Coordinate system $\mathbb{R}^3$ with coordinates (x, y, z)
η	[kg/ms]	Molecular viscosity
ν	[m^2/s]	Kinematic viscosity, μ/ρ
ν_ε	[m^2/s]	Isotropic eddy viscosity
ν_t	[m^2/s]	Turbulent kinematic viscosity $\nu_t = \nu_0 + c_\mu u^* h$
ν_0	[m^2/s]	Base kinematics eddy viscosity
ρ	[kg/m^3]	Mass density (fluid)
ρ_s	[kg/m^3]	Bed material density
$\tau_{B,x}, \tau_{B,y}$	[N/m^2]	Cartesian components of bottom shear stress vector
$\boldsymbol{\tau}_B$	[N/m^2]	Vector of shear stress at bottom due to water flow
$\tau_{B_g,crit}$	[N/m^2]	Critical shear stress related to grain size d_g
$\tau_{S,x}, \tau_{S,y}$	[N/m^2]	Cartesian components of surface shear stress vector
$\boldsymbol{\tau}_S$	[N/m^2]	Vector of shear stress at water surface (e.g. due to wind)
Ω	[m^2]	Area of an element
ξ_g	[-]	Hiding factor

This page has been intentionally left blank.